David Schultz

Il libro del Commodore VIC 20

seconda ristampa

Il piacere del computer

Il piacere del computer serie diretta da Mauro Boscarol

1 *Tom Rugg e Phil Feldman* 32 programmi con il PET
2 *Rich Didday* Intervista sul personal computer, hardware
3 *Tom Rugg e Phil Feldman* 32 programmi con l'Apple
4 *Ken Knecht* Microsoft Basic
5 *Paul M. Chirlian* Pascal
6 *Tom Rugg e Phil Feldman* 32 programmi con il TRS-80
7 *Rich Didday* Intervista sul personal computer, software
8 *Herbert D. Peckham* Imparate il Basic con il PET/CBM
9 *Karl Townsend e Merl Miller* Il personal computer come professione
10 *Karen Billings e David Moursund* Te ne intendi di computer?
11 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, uno: introduzione
12 *Don Inman e Kurt Inman* Imparate il linguaggio dell'Apple
13 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, due: applicazioni
14 *Luigi Pierro* Il manuale del CP/M
15 *Carlo Sintini* A scuola con il PET/CBM
16 *David Johnson-Davies* Il manuale dell'Atom
17 *David E. Schultz* Il libro del Commodore VIC 20

David Schultz

Il libro del Commodore VIC 20

franco muzzio & c. editore

Indice generale

CAPITOLO UNO

Introduzione

Questo libro fornisce una serie di esercizi per insegnare l'uso del VIC Commodore ai suoi possessori. Il metodo più efficace per usare questo libro è quello di sedersi di fronte al VIC e svolgere fedelmente gli esercizi proposti. Inoltre, nel libro vi è spazio sufficiente per consentire all'utente di inserire le proprie risposte. Quando poi occorrerà rinfrescare la memoria su di un punto particolare, questi esempi forniranno dati utili.

1.1 CONOSCENZE DA PARTE DELL'UTENTE

Forse qualche utente avrà dimestichezza con la matematica. Tuttavia, questo libro presuppone che la maggioranza non abbia particolari cognizioni matematiche. Per questa ragione, la maggior parte degli esercizi adopererà la semplice aritmetica.
Questo libro presuppone anche una certa conoscenza del vocabolario e della sintassi del linguaggio di programmazione Basic. Se così non fosse, i testi sottoelencati potranno risultare utili:

DIDDAY, *Intervista sul personal computer, hardware*
DIDDAY *Intervista sul personal computer, software*

DWYER, CRITCHFIELD *Il Basic e il personal computer. Uno: introduzione*
DWYER, CRITCHFIELD *Il Basic e il personal computer. Due: applicazioni*
KNECHT *Microsoft Basic*

tutti apparsi in questa collana.

Ricordatevi di alternare la lettura di questo libro e del vostro manuale di Basic: otterrete così il doppio vantaggio di imparare il Basic e l'uso del VIC.

1.2 ESERCIZI

Alcuni esercizi sono stati studiati per essere eseguiti meccanicamente. Sono, cioè, del tipo "botta e risposta". Mentre si impara un certo argomento, gli esercizi assumono la forma: "Cosa succede se si batte questo?". Il successivo livello suggerisce che si provino varie risposte.
Se avete capito l'argomento, sarete in grado di anticipare la maggior parte delle risposte. Ove vi siano dei concetti particolarmente difficili, delle eccezioni o dei problemi particolari, si incontreranno esercizi del tipo "prova questo...".
L'ultimo livello di esercizi è del tipo "fatelo da voi", nei quali vengono forniti degli spunti per creare il proprio esempio. Se non siete in grado di costruire il vostro esempio, occorrerà ripassare la lezione e riprovare. Scoprirete di imparare di più provando e riprovando nuovi argomenti. Assicuratevi di scrivere sul libro le prove effettuate, giuste o sbagliate che siano, in modo da avere un riferimento su ciò che è stato fatto. Eseguite tutti gli esercizi del libro. Non saltatene alcuno.

1.3 PROGRAMMAZIONE

Contrariamente a quanto qualcuno afferma, il modo migliore per imparare a programmare è quello di... programmare. Si può acquisire la tecnica della programmazione leggendo i libri sull'argomento, ma ciò è simile alla lettura delle mosse nel gioco degli scacchi. Conoscere le regole degli scacchi non implica saper giocare a scacchi. Conoscere la sintassi del Basic non implica saper programmare bene in Basic.
Come si impara la strategia nel gioco degli scacchi osservando le mosse di un buon giocatore, così si impara a programmare leggendo i programmi fatti da un buon programmatore. Ma attenzione: la lettura di

un listato mal fatto insegna cose errate, così come la lettura di un listato ben fatto insegna cose giuste.
Vi sono molte regole utili che tutti i programmatori dovrebbero osservare. Innanzitutto, se si vuole avere successo nella programmazione occorre acquisire un metodo coerente. Avere successo nella programmazione significa essere capaci di scrivere velocemente ed accuratamente lunghi listati facili da correggere ed usare. Perché è importante acquisire un metodo coerente? Se ci si accosta a problemi diversi con la stessa metodica è possibile ottenere i seguenti risultati:

1. Usare parti di programma altrove. Ciò risparmia tempo poiché non è necessario riaffrontare lo stesso problema.
2. Comprendere facilmente ciò che è stato scritto in passato. Il metodo rigoroso minimizza domande come: "A cosa pensavo quando ho affrontato questo problema?".
3. Riusare documentazione di precedenti programmi.
4. Usare con coerenza messaggi di errore.

In secondo luogo, è opportuno seguire un certo piano prima di battere il listato. Questo fa certamente parte del vostro metodo o stile di risoluzione dei problemi. Una tentazione alla quale molti programmatori cedono è quella di agire d'impulso, cioè sedersi di fronte alla tastiera ed iniziare a battere il programma senza pensarci troppo. Non fatelo! Neanche quando si sa che il problema in questione è banale. Il pericolo nel battere senza ragionare su di un problema banale è che il programma può funzionare anche bene. Ma ad un certo punto, ci si potrebbe imbattere in un programma solo all'apparenza banale, ma che, invece, non lo è affatto. Tutti i precedenti successi nella programmazione d'impulso renderanno difficile ammettere che il problema non è poi tanto banale. Inoltre, tutti i precedenti successi non aiuteranno a formare una strategia nello sviluppare i programmi, ma anzi, vi sarà sempre la tentazione di utilizzare questo metodo qualora si creda di dover affrontare un problema banale. La fig. 1 indica un buon approccio per scrivere dei programmi.
Qual è il metodo migliore per ottimizzare un programma? Quando il programma è ultimato, cioè quando questo funziona a dovere, trascuratelo e tornate alla fase di studio. Perché agire così? Perché una volta conosciuti i risultati forniti dal programma, si individuano meglio i suoi difetti e la fase di studio sarà migliore e più concisa. Cercare di ottimizzare un programma tappando le falle, d'altra parte, può introdurre errori indesiderati. Ricordatevi che non è mai troppo tardi per scartare un programma e ricominciare dall'inizio. Occorre osservare il programma come se fosse una falsa riga di esecuzione, qualcosa da cui si può imparare. Anche se la sua vita è stata breve, è servito a migliorare.
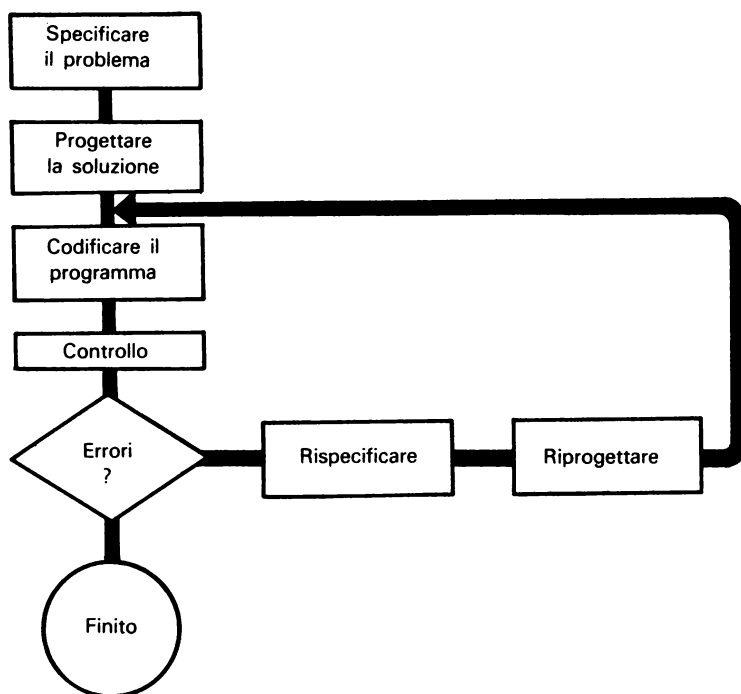
Fig. 1 Diagramma di flusso per lo sviluppo di un programma

Molte pubblicazioni insistono sul concetto che la programmazione è facile e chiunque può imparare a programmare. Questo non è vero. Durante gli ultimi anni, abbiamo insegnato a programmare in diversi linguaggi e su diverse macchine a molte persone di cultura diversa. Qualcuno è diventato programmatore, mentre molti altri non vi sono riusciti. La nostra esperienza non è unica. Così come qualcuno può diventare un bravo giocatore di bridge o di tennis, oppure un buon pianista, altri non lo diventano e non lo possono diventare.
Tuttavia non è necessario diventare bravissimi programmatori per divertirsi con il VIC. Sono disponibili nastri magnetici a cassetta contenenti programmi. Se si conosce il linguaggio Basic è possibile ristrutturare questi programmi, personalizzandoli secondo le vostre esigenze.

1.4 LA TASTIERA ED IL VIDEO DEL VIC

La tastiera ed il video del VIC sono i principali organi di ingresso e uscita (input, output). Per comunicare con successo con il vostro VIC, bisogna conoscere i caratteri presenti sulla tastiera e sapere come utilizzare il video. Questo si può imparare svolgendo gli esercizi del manuale fornito a corredo della macchina.

Fig. 2 La tastiera del VIC.

1.5 RIPASSO DEL BASIC

Il Basic ha tre tipi di istruzioni: comandi, dichiarazioni e funzioni.
I *comandi* sono delle istruzioni che possono essere considerate parte del sistema operativo. I comandi operano su intere strutture come ad esempio i programmi.
Le *dichiarazioni* sono elementi propri di un programma. Esse sono composte da parole chiave, variabili, costanti, operatori e funzioni.
La *funzione* definita dall'utente è una dichiarazione Basic che restituisce un singolo valore ogniqualvolta viene invocata. Un altro tipo di funzione è quella di biblioteca. Il VIC possiede diverse funzioni di biblioteca, come SIN, RND, LOG, INT, ecc. Queste funzioni sono richiamate con il loro nome.

1. *Costanti* I programmi Basic sono composti da dichiarazioni contenenti parole chiave, variabili, operatori e costanti. Il VIC può gestire due tipi di costanti. Le costanti numeriche sono numeri con o senza punto decimale. Se un numero ha il punto decimale, è del tipo in virgola mobile. Un numero senza punto decimale è detto intero. La virgola mobile viene usata per la rappresentazione di un insieme di numeri più vasto di quello degli interi (vedi cap. 6 sulla rappresentazione dei dati per ulteriori dettagli). Anche se esistono due tipi di costanti numeriche, il VIC le rappresenta nello stesso modo, cioè con la virgola mobile. Una costante intera, come 5, viene rappresentata come 5.0. L'altro tipo di costante che il VIC gestisce è di tipo stringa od alfanumerica. Una costante stringa è individuata dalle virgolette (" "). I simboli "AB" indicano una stringa di due caratteri, la A seguita dalla B.

2. *Variabili* Le variabili sono "nomi" che possono assumere valori diversi nel corso di un problema. Nel Basic il primo carattere di una variabile deve essere alfabetico e non numerico. 1 A=2 indica al computer che il contenuto della linea 1 è A=2.

ESERCIZIO Sperimentate diversi tipi di caratteri consentiti per i nomi delle variabili.

```
A= 1
READY.
AB= 2
READY.
ABC= 3
READY.
ABCABCABC= 4
READY.
A12345678= 5
READY.
A123456789123456789= 6
READY.
```

Sebbene il primo carattere debba essere alfabetico, quelli seguenti possono essere sia che alfabetici che numerici, assumendo che il nome della variabile può essere di lunghezza qualsiasi. Infatti, lunghezza a parte, solo i primi due caratteri sono utilizzati per distinguere una variabile dall'altra.

ESERCIZIO Verificate che nei nomi di variabili sono utilizzati solo i primi due caratteri.

```
PRINT A1
6
READY.
```

Si noti che le variabili A123456789123456789, A12345678 e A1 si riferiscono tutte alla stessa quantità. Verificate ciò visualizzando i tre valori.

```
PRINT A123456789123456789; A12345678; A1
```

(NOTA Sebbene venga visualizzato tutto su di un'unica linea, il video del VIC può gestire solo 23 caratteri per linea, per cui dividerà l'istruzione PRINT. Ricordatevelo.)

```
 6  6  6
READY.
```

I primi due caratteri sono gli unici utilizzati, tuttavia gli altri possono essere usati come descrittivi della variabile.

```
ABCONTEGGIO=33
READY.
PRINT AB
33
READY.
```

ABCONTEGGIO potrebbe indicare il conteggio delle variabili A e B.

CAPITOLO DUE

Calcolare con il Basic

Il modo "*calcolatore*" indica le istruzioni che vengono eseguite immediatamente. Queste istruzioni non possiedono il numero di linea. La mancanza del numero di linea segnala al sistema operativo che queste istruzioni devono essere eseguite immediatamente e non devono essere memorizzate nel programma. Il modo calcolatore è stato utilizzato per la determinazione dei nomi leciti per le variabili nei precedenti esempi. Quando si battono delle linee, ricordatevi che ogni linea deve terminare premendo il tasto RETURN.

```
A=5
READY.
B=10
READY.
C=A*B
READY.
```

Queste tre linee non hanno avuto alcun effetto visibile sul VIC. Tuttavia, le tre variabili A, B e C sono state inizializzate con i valori indicati. Per assicurarvi che il contenuto di C sia stato ben calcolato, battete:

```
PRINT C
50
READY.
```

Per controllare tutte e tre le variabili in una volta:

```
PRINT A;B;C
5  10  50
READY.
```

Le variabili nel listato in uscita sono separate da due spazi.
Il modo calcolatore può essere utilizzato per calcolare valori direttamente con l'istruzione PRINT.

```
PRINT 5*10
50
READY.
PRINT A*B+C
100
READY.
```

Un'abbreviazione dell'istruzione PRINT nel Basic Commodore è ?.

```
?5*10
50
READY.
```

2.1 L'UTILIZZAZIONE DELLE STRINGHE NEL BASIC COMMODORE

D'ora in poi non verrà più indicata la risposta READY del computer, che sarà sottintesa. Le virgolette sono usate per definire i limiti di una stringa di caratteri. Battete qualche stringa.

```
A$="NOME E'"
PRINT A$
NOME E'
B$="IL MIO"
PRINT B$+A$
IL MIONOME E'
```

Il segno + concatena le due stringhe A$ e B$ solo per il video. Concatenare significa congiungere. Non vi è spazio tra IL MIO e NOME E'. Lo aggiungiamo noi.

```
PRINT B$+" "+A$
IL MIO NOME E'
B$=" IL MIO "
PRINT B$+A$
 IL MIO NOME E'
```

Bisogna essere prudenti quando si batte $+. Se non viene rilasciato il tasto SHIFT prima della battitura, si avrà un ?SYNTAX ERROR. La linea sembrerà andar bene, poiché premendo contemporaneamente i tasti SHIFT e + si ottiene un carattere grafico simile ad una croce. Per salvare la stringa concatenata:

```
C$=B$+A$
PRINT C$+" DAVIDE"
IL MIO NOME E' DAVIDE
```

2.2 VALORI NUMERICI E FRAZIONARI

Battete qualche numero per comprendere le capacità numeriche del VIC.

```
X=.25
Y=10
Z=X*Y
PRINT Z
2.5
? Y/Z
40
```

Si può usare un ? al posto del PRINT. Il simbolo / indica una divisione, mentre una freccia verso l'alto ↑ indica un'elevazione a potenza.

```
? Y↑3
1000
PRINT Z
2.5
PRINT INT(Z)
2
```

INT fornisce la sola parte intera del numero.

2.3 CONVERSIONE DI DATI

Esistono diverse funzioni disponibili nel Basic Commodore per la conversione da una forma di rappresentazione dei dati ad un'altra. È già stato visto INT che fornisce la parte intera di un numero.

```
PRINT ASC("Z")
90
```

ASC restituisce il numero di codice ASCII relativo ad un carattere; CHR$ restituisce un carattere dato il relativo numero di codice ASCII.

```
L=ASC("Z")
PRINT L; CHR$(L)
90 Z
L=ASC("ZXCV")
?L
90
```

Il numero 90 corrisponde al carattere Z. ASC restituisce solo il codice del primo carattere di una stringa.

```
PRINT CHR$(256)
?ILLEGAL QUANTITY
 ERROR
```

I caratteri consentiti vanno da 0 a 255.

```
PRINT CHR$(255)
π
```

Il numero 255 corrisponde al carattere pi greco (π).

ESERCIZIO Qual è il codice ASCII di "?"? Qual è il carattere corrispondente al numero 71?

____

____

____

____

2.4 UTILIZZO DEL MODO CALCOLATORE NELLA GESTIONE DEL CONTO CORRENTE

Il VIC può essere di aiuto nella gestione del conto corrente. Rappresentate con la variabile SP il saldo precedente.

```
SP=990250
SP=SP-12360
PRINT SP
977890
```

Il saldo, dopo aver sottratto un assegno di 12360 lire, è di 977890.

```
SP=SP-10000-20000-22500-11980
?SP
913410
```

Questo è il saldo dopo aver sottratto altri 4 assegni.

2.5 PAROLE RISERVATE

Il Basic Commodore riserva l'uso di certe stringhe di caratteri per specificare le operazioni da svolgere.

ESERCIZIO Determinate ciò che fa il VIC usando impropriamente delle parole riservate.

```
INT=3
?SYNTAX
 ERROR
```

Poiché INT indica la funzione "parte intera", non può essere usata come variabile. Cosa vi aspettate da:

```
ASC=1
CHR$=1
PRINT=12
```

2.6 TIPI DI VARIABILI E COSTANTI

Vi sono due tipi di variabili e di costanti: stringhe di caratteri e numeri in virgola mobile. Non è consentito confondere tra di loro stringhe e numeri.

ESERCIZIO Non confondete stringhe e numeri.

```
A=CHR$(63)
?TYPE MISMATCH
 ERROR
```

È stato dato un messaggio di errore poiché A è una variabile numerica, mentre CHR$ richiede un carattere.

ESERCIZIO Visualizzate l'equivalente numerico del carattere "?".

```
A=ASC("?")
PRINT A
63
```

Cosa otterrete con:

```
D$=CHR$(63)
PRINT D$
```

CAPITOLO TRE

Introdurre un programma

Ogni linea di programma inizia con un numero. Invece qualsiasi istruzione introdotta senza numero di linea viene eseguita immediatamente. Questo è il modo calcolatore già utilizzato nel capitolo 2. Usando il Basic, qualsiasi linea di testo che inizi con una cifra viene elaborata solo dopo aver battuto RUN. Vi sono quattro possibilità diverse:

1. Viene aggiunta al programma una nuova linea. Questo succede se il numero di linea è lecito (da 0 a 63999 per il VIC).

2. Viene modificata una linea preesistente. Questo succede quando il numero di linea coincide con il numero di una linea già esistente nel programma. Quest'ultima viene sostituita dalla nuova linea introdotta.

ESERCIZIO Utilizzate questa caratteristica per cambiare un'istruzione nel vostro programma Basic.

```
100 A=B+C
```

Se volete cambiare la B in D nella linea 100, un modo è il seguente:

```
100 A=D+C
```

3. Viene cancellata una linea preesistente quando viene introdotto il

numero di una linea già esistente nel programma. La linea deve contenere il solo numero di linea; altrimenti la linea verrà modificata, ma non cancellata.

4. Viene generato un messaggio di errore (?SYNTAX ERROR) se il numero di linea è esterno all'intervallo consentito (da 0 a 63999).

3.1 SPAZI VUOTI

Gli spazi vuoti che precedono un numero di linea vengono ignorati. Il primo carattere di una linea che non è una cifra pone termine al numero di linea. Spazi vuoti sono permessi ovunque in una linea per migliorare la lettura, purché non interni a parole riservate (ad esempio PRINT) o a costanti numeriche. Poiché il Basic ignora spazi tra variabili, operatori e parole chiave delle istruzioni (a meno che non siano tra virgolette), non ha importanza se battete PRINT A ; B oppure PRINTA;B. In questo libro si è cercato di seguire una certa regolarità nella spaziatura, ma noterete delle variazioni.

3.2 ISTRUZIONI MULTIPLE IN UN PROGRAMMA

Su una singola linea possono apparire più istruzioni separate da ":". Il numero di linea può esserci *solo* all'inizio della prima istruzione della linea. Si può ottenere un risparmio di memoria eliminando le virgolette finali nelle stringhe, visto che le virgolette richiedono un byte di memoria. Per esempio, PRINT "ABC funziona altrettanto bene come PRINT "ABC", purché la stringa di caratteri sia l'ultima parte della linea. Perciò si dovranno usare virgolette finali nel seguente caso: 100 PRINT "A=",X.
Per evitare di ritrovarsi con programmi troppo difficili da leggere e correggere, usate le virgolette anche se si spreca un byte.

Ecco qualche esempio di linee con più istruzioni.

```
NEW
10 PRINT"AAA:PRINT"BBB:PRINT"CCC
RUN
AAA:PRINT    0
CCC
```

Il sistema visualizza AAA:PRINT come una stringa di caratteri visto che è racchiusa tra virgolette. Poi assume che BBB sia una variabile e visualizza il valore di BBB che è zero. PRINT "CCC, che è la successiva istruzione, viene invece interpretata correttamente.
L'esempio riportato mostra qualcuno dei problemi che si possono incontrare quando viene tralasciata una virgoletta.

ESERCIZIO Uso appropriato delle virgolette in linee con più istruzioni.

```
10 PRINT"AAA":PRINT"BBB":PRINT"CCC"
RUN
AAA
BBB
CCC

10 A$="AB"
20 PRINT A$
RUN
AB
```

ESERCIZIO Anche nel modo calcolatore sono permesse linee con più istruzioni.

```
FOR I=1 TO 20: PRINT I: NEXT I
```

3.3 ERRORI DI BATTITURA

Se si fanno errori di battitura durante l'introduzione di una linea, vi sono tre modi per correggere l'errore:

1. Battere il tasto DEL fino a cancellare i caratteri voluti. Ribattere il resto della linea da quel punto.

2. Battere il tasto RETURN e ribattere l'intera linea.

3. Battere il tasto CLR per pulire lo schermo e non introdurre tale linea in memoria.

3.4 COMANDI DEL BASIC COMMODORE

LIST numero di linea—numero di linea

Per visualizzare il vostro programma si usa il comando LIST (vedere fig. 3). La visualizzazione avverrà in ordine crescente di numero di linea, a prescindere dall'ordine con cui le linee sono state introdotte. Tra il numero di linea ed il primo carattere non numerico della linea, viene inserito automaticamente uno spazio, anche se non è stato originariamente introdotto.
Se viene dato il comando LIST senza numero di linea, verranno visualizzate tutte le istruzioni in memoria.
Se viene dato il comando LIST 100, verrà visualizzata solo la linea 100.
Se viene dato il comando LIST 100-150, verranno visualizzate tutte le istruzioni tra 100 e 150.
Se viene dato il comando LIST 100-, verranno visualizzate tutte le istruzioni da 100 in poi.
Se viene dato il comando LIST-100, verranno visualizzate tutte le istruzioni tra 0 e 100.
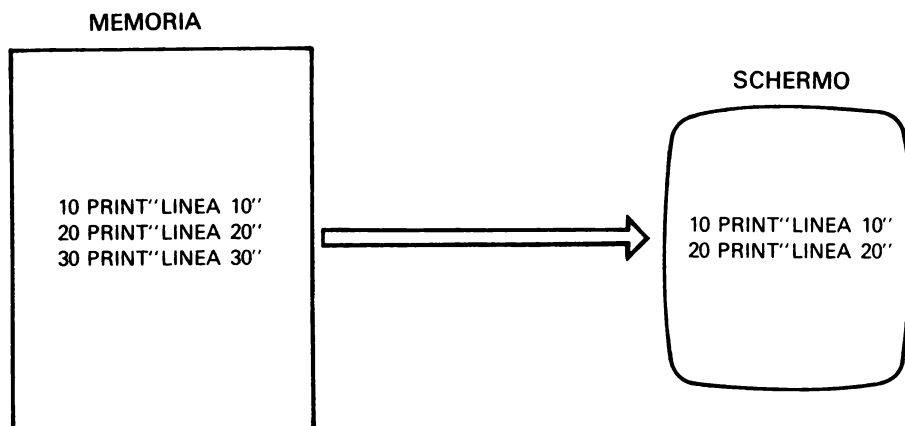
Fig. 3 Funzione del comando LIST.

ESERCIZIO Uso delle varie opzioni del comando LIST.

```
NEW
10 PRINT"LINEA 10"
30 PRINT"LINEA 30"
20 PRINT"LINEA 20"
LIST
10 PRINT"LINEA 10"
```

```
20 PRINT"LINEA 20"
30 PRINT"LINEA 30"
LIST -20
10 PRINT"LINEA 10"
20 PRINT"LINEA 20"
LIST 20-
20 PRINT"LINEA 20"
30 PRINT"LINEA 30"
```

LIST-20 visualizza tutte le istruzioni fino alla linea 20 compresa.
LIST 20- visualizza tutte le istruzioni dalla linea 20 compresa in poi.

NEW

Il comando NEW viene utilizzato per cancellare il programma in memoria prima di inserirne uno nuovo. Se non viene dato il comando NEW e si inizia ad inserire un nuovo programma, il sistema non riesce a distinguere se si tratta di un nuovo programma o di variazioni al programma precedente. Se volete memorizzare il vecchio programma, assicuratevi di iniziare il nuovo programma con un numero di linea più grande del maggiore numero di linea di quello vecchio.

ESERCIZIO Dimostrate che il NEW cancella dalla memoria il programma.

```
NEW
10 REM NUOVO TEST
LIST
10 REM NUOVO TEST
NEW
LIST
READY.
```

La prova dell'efficacia del comando NEW consiste nell'usare il comando LIST.

RUN numero di linea

Il numero di linea è opzionale; se viene trascurato, l'esecuzione del programma comincia dal numero di linea più basso e continua finché non è stata eseguita l'ultima istruzione del programma oppure non si incontrano le istruzioni END o STOP. Se invece si batte RUN seguito dal numero di linea, l'esecuzione parte da quella linea.

ESERCIZIO Controllo dell'esecuzione di un programma con RUN.

```
NEW
10 PRINT"LINEA 10"
20 PRINT"LINEA 20"
30 PRINT"LINEA 30"
40 END
50 PRINT"LINEA 50"
60 PRINT"LINEA 60"
RUN
LINEA 10
LINEA 20
LINEA 30
```

RUN comincia l'esecuzione dal numero di linea più basso e continua finché non si incontrano le istruzioni END o STOP, oppure non è stata eseguita l'ultima istruzione.

```
RUN 50
LINEA 50
LINEA 60
```

RUN 50 fa cominciare l'esecuzione dalla linea 50 e continua fino a quando non ci sono altre istruzioni.

```
RUN 20
LINEA 20
LINEA 30
```

RUN 20 fa cominciare l'esecuzione dalla linea 20 e si ferma quando si incontra l'istruzione END.

ESERCIZIO Esempio dí due programmi in memoria contemporaneamente.

```
NEW
10 R=3
20 A=π*R*R
30 PRINT A
40 END
```

Questo primo programma (10–40) calcola l'area di una circonferenza di raggio 3.

```
100 X=π/2
110 A=SIN(X)
120 PRINT A
130 END
```

Il secondo programma (100-130) calcola il seno di 90 gradi. Coesistono in memoria due programmi distinti e mostreremo l'uso di ciascuno.

ESERCIZIO Esecuzione selettiva di uno dei due programmi in memoria. I programmi dell'esercizio precedente devono essere ancora in memoria.

```
RUN
28.27343339
```

Questo programma visualizza l'area di una circonferenza il cui raggio è stato battuto nella linea 10.

3.5 STOP E CONT

Quando un programma si trova in un ciclo senza fine, premendo il tasto STOP si causa l'interruzione dell'esecuzione e si osserva sul video:

BREAK IN numero di linea

Il numero di linea dipenderà dall'istante in cui è stato premuto lo STOP. Battendo CONT l'esecuzione riprenderà da quel punto.

ESERCIZIO Uso di STOP e CONT. Introdurre il seguente programma:

```
NEW
10 I=0
20 FOR A=1 TO 500
30 X=A
40 NEXT A
44 I=I+1
46 PRINT I
50 GO TO 20
RUN
1
2
```

```
'STOP'
BREAK IN 30
CONT
```

Se il vostro programma è in attesa di un input da tastiera, lo STOP non provocherà un BREAK. Quando si è in queste condizioni, il tasto RETURN dopo il tasto STOP fermerà il programma.

CAPITOLO QUATTRO

Ricavare i dati dal programma

Un programma può calcolare diversi valori. A meno che non si abbia un modo di vedere i risultati, questi calcoli sono inutili. Si osservi la fig. 4 per comprendere il modo di visualizzare i dati calcolati. I risultati possono essere visualizzati su di uno schermo in diversi modi.

4.1 FORMATI DI USCITA PER DATI NUMERICI

ESERCIZIO Uscita di variabili non inizializzate in formato "*campo*". Le variabili non inizializzate sono quelle che non sono state ancora usate. Il Basic pone tali variabili automaticamente a zero.

```
NEW
PRINT A,B,C,D,E
 0          0
 0          0
 0
```

I campi numerici hanno ciascuno una larghezza di 11 colonne ed iniziano nelle seguenti colonne. Ammesso che la prima colonna sia contrassegnata con 1:

campo	*colonna*
1	1
2	12
3	1 della seconda riga
4	12 della seconda riga
5	1 della terza riga

ESERCIZIO Sperimentate i differenti modi con cui si visualizzano i numeri (i due punti servono a delimitare più istruzioni sulla stessa linea).

```
NEW
100 A=-1: B=-2
110 PRINT A,B
120 A=1: B=2
130 PRINT A,B
140 A=-111: B=-222
150 PRINT A,B
160 A=111: B=222
170 PRINT A,B
RUN
-1         -2
 1          2
-111       -222
 111        222
```

Si noti che all'inizio di ogni campo numerico è riservato uno spazio per il segno del numero (+/-). Tuttavia, quando si tratta di numeri positivi il segno non viene stampato; perciò i numeri positivi hanno inizio alle colonne 2 e 13.
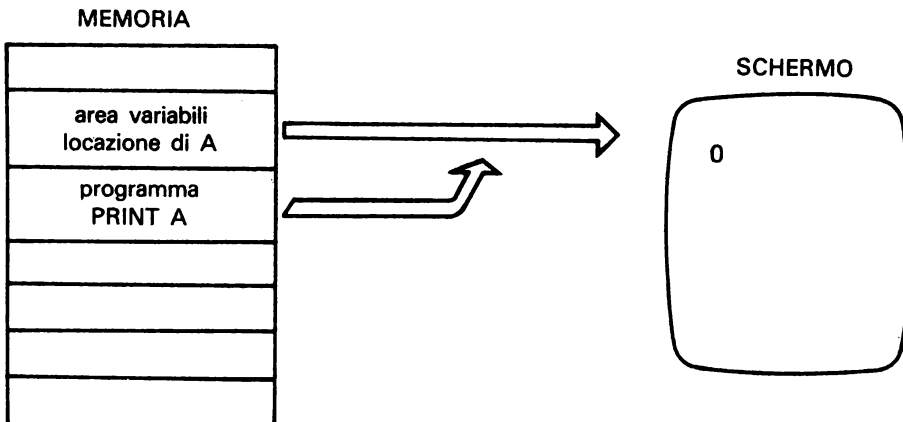
Fig. 4 Funzione del PRINT.

ESERCIZIO Provate a battere diversi esempi di numeri separati dalla virgola.

```
PRINT 1,2,3,4,5,6,7,8,9,10,11,12
 1              2
 3              4
 5              6
 7              8
 9              10
 11             12
```

ESERCIZIO Visualizzazione di variabili in formato continuo.

```
A=1: B=2: C=3
PRINT A;B;C
 1  2  3
```

Quando, per la separazione degli elementi di una istruzione PRINT, viene usato il punto e virgola ogni numero positivo dista dal successivo di due spazi. Il primo numero è preceduto da uno spazio.

ESERCIZIO Sperimentate, con i diversi tipi di numeri, la spaziatura con il punto e virgola.

```
NEW
100 A= -1: B= -2: C= -3
110 PRINT A;B;C
120 A=1: B=2: C=3
130 PRINT A;B;C
140 A=-111: B=-222: C=-333
150 PRINT A;B;C
160 A=111: B=222: C=333
170 PRINT A;B;C
RUN
-1 -2 -3
 1  2  3
-111 -222 -333
 111  222  333
```

Si noti che, come nel caso della spaziatura con virgola, davanti al numero è riservato uno spazio per il segno. In più, ogni numero è seguito da uno spazio. Perciò i numeri negativi sono separati da uno spazio, quelli positivi da due (uno per la spaziatura ed uno per il segno).

ESERCIZIO Visualizzate vari valori che esemplifichino le tecniche di stampa.

```
NEW
100 A=1
110 FOR I=1 TO 4
120 A=A*10
130 PRINT A,A
140 PRINT A;A
150 PRINT
160 NEXT I
RUN
```

Si noti come vengono visualizzati i numeri.

4.2 FORMATI DI USCITA PER STRINGHE DI CARATTERI

ESERCIZIO Uscita di stringhe in campi e formati continui.

```
NEW
100 A$="A": B$="B": C$="C": D$="D"
105 E$="E": F$="F": G$="G"
110 PRINT A$,B$,C$,D$,E$,F$,G$
120 PRINT A$;B$;C$;D$;E$;F$;G$
RUN
A          B
C          D
E          F
G
ABCDEFG
```

Una virgola nell'istruzione PRINT dispone le stringhe in campi successivi (distanziati di 11 colonne). Si noti che non vi è lo spazio per il segno come nelle variabili numeriche.
Se nell'istruzione PRINT vi è un punto e virgola le stringhe vengono visualizzate una dopo l'altra, senza spazi.

Ricapitolando le convenzioni di punteggiatura proprie dell'istruzione PRINT sia per i numeri che per le stringhe, il punto e virgola riserva uno spazio per il segno del numero, che rimarrà vuoto se il segno è positivo. Inoltre i numeri sono separati da uno spazio. Le stringhe vengono visualizzate unite senza spazi. La virgola utilizza campi di 11 caratteri. Le

stringhe sono visualizzate all'inizio di ogni campo; con i numeri il primo spazio del campo è riservato al segno del numero.

4.3 SPAZIATURA

ESERCIZIO Utilizzate la funzione SPC per visualizzare degli spazi.

```
NEW
200 FOR I=0 TO 22
210 PRINT SPC(I);"A"
220 NEXT I
RUN
```

La funzione SPC permette di visualizzare un numero variabile di spazi prima dei dati specificati. Verrà visualizzata una serie di A disposte in diagonale. Se la visualizzazione è troppo veloce e non permette il controllo della posizione di ogni A, si prema il tasto CTRL. Questo rallenterà la visualizzazione.

ESERCIZIO Utilizzate la funzione TAB per incolonnare i dati sul video:

```
210 PRINT TAB(I);"A"
RUN
```

La funzione TAB visualizzerà i dati a partire dalla colonna indicata dalla variabile I. La differenza tra la funzione TAB ed SPC è che la prima indica la colonna di inizio, mentre la seconda indica il numero di spazi da interporre tra i campi. Si noti che la prima colonna è 0 e non 1.

ESERCIZIO Mostrate la differenza tra TAB e SPC.

```
NEW
10 FOR I=2 TO 10 STEP 2
20 PRINT SPC(I);"A";SPC(I);"B"
30 PRINT TAB(I);"C";TAB(I+2);"D"
40 PRINT
50 NEXT I
RUN
```

La linea 20 stampa I spazi (I essendo diverso in ogni ciclo) da interporre tra la A e la B. La linea 30 restituisce costantemente uno spazio tra C e D.

CAPITOLO CINQUE

Introdurre informazioni nel programma

5.1 OBIETTIVI DA RAGGIUNGERE

Uno degli obiettivi da raggiungere è quello di minimizzare la quantità ed il tipo di dati che l'utente deve introdurre nel corso del programma. I dati richiesti dovranno essere brevi, semplici e logici. Un maggior numero di dati in ingresso corrisponde ad un maggior spreco di tempo ed anche a maggiore probabilità di commettere errori.
Nello scrivere un programma, una buona regola da tener presente è quella di non utilizzare mai una costante ove vi sia la possibilità che questa, più tardi, possa essere cambiata. In quei casi si utilizzerà una variabile al posto della costante.
In apparenza, queste due filosofie sembrano essere in conflitto tra di loro. Tuttavia, il Basic fornisce molti differenti metodi per arricchire il programma con dati, e cioè assegnando alle variabili particolari valori. Le parole chiave per arricchire il programma con dati sono

INPUT, READ e DATA

5.2 INPUT

INPUT consente all'utente di fornire al programma dati da elaborare. Questi possono essere o dati di controllo per la selezione di opzioni e per specificare come eseguire l'elaborazione, o dati da elaborare, o ambedue. È possibile visualizzare un messaggio contemporaneamente alla richiesta di input, racchiudendolo tra virgolette e facendolo seguire all'istruzione INPUT.

ESERCIZIO Mostrate come è possibile, a richiesta, introdurre dei dati.

```
NEW
10 INPUT"QUAL E' IL TUO NUMERO";N
20 ? N
30 IF N<>0 THEN 10
RUN
QUAL E' IL TUO NUMERO?
5
QUAL E' IL TUO NUMERO?
0
READY.
```

Il messaggio deve essere breve. Se è più lungo di 23 caratteri, verrà considerato come un dato.

ESERCIZIO Dimostrate il limite di lunghezza dei messaggi.

```
NEW
100 INPUT"QUESTO E' UN
PROMPT MOLTO LUNGO";I
110 PRINT I
RUN
QUESTO E' UN PROMPT MO
LTO LUNGO?
5
?REDO FROM START
QUESTO E' UN PROMPT MO
LTO LUNGO?
'STOP' 'RESTORE'
```

Si osservi ciò che il computer ha considerato valido come input.

```
100 INPUT "QUESTO E' UN PROMPT MOLTO LUNGO";I$
110 PRINT I$
RUN
QUESTO E' UN PROMPT MO
LTO LUNGO?
5
QUESTO E' UN PROMPT MO
LTO LUNGO?5
READY.
```

Ciò dimostra che il computer ha considerato valido come input il 5 preceduto dal messaggio. La soluzione al problema è di contenere il messaggio in una linea.
L'INPUT viene utilizzato per la lettura dei dati dalla tastiera del VIC (vedere fig. 5). Quando viene utilizzata l'istruzione INPUT, viene visualizzato sullo schermo un ?.

1. INPUT *in virgola mobile* Questo tipo di INPUT permette di inserire nel VIC numeri molto grandi o molto piccoli.

ESERCIZIO Programma per il controllo della validità dei dati in virgola mobile.

```
NEW
4000 INPUT V:PRINT V:GO TO 4000
RUN
?
```
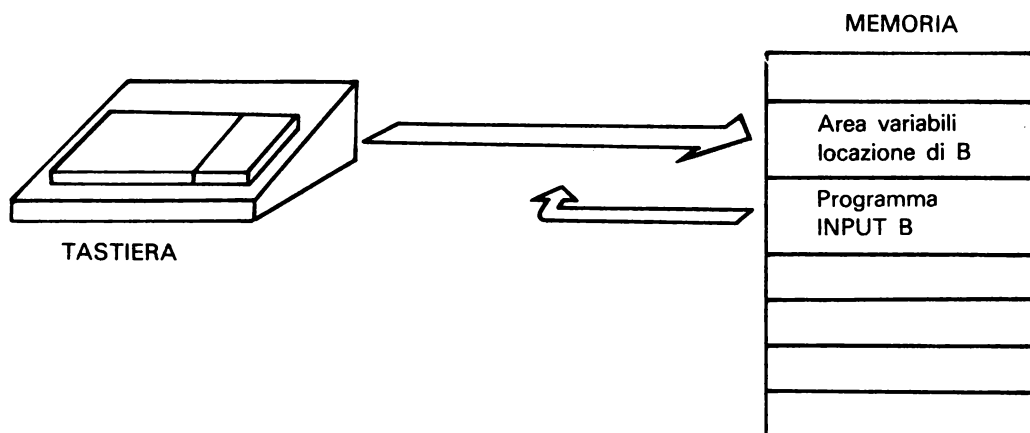
Fig. 5 Funzionamento dell'INPUT

Siccome V è una variabile in virgola mobile, si può utilizzare questa parte di programma per sperimentare il formato o la grandezza dei numeri che il Basic Commodore può gestire. Il ? significa che il programma attende l'ingresso dei dati.

```
1.5
1.5
?
```

È possibile battere numeri frazionari.

```
1.5E+5
150000
?
```

Per numeri molto grandi si può utilizzare la notazione scientifica.

```
.005E+6
5000
?
1235.01E+2
123501
```

La notazione scientifica può anche essere usata per spostare il punto decimale in un numero.

```
?
12.5E+40
?OVERFLOW
 ERROR IN 4000
```

Questo messaggio significa che il numero 12.5E+40 è troppo grande. Prima si è visto che 1.5E+5 non era troppo grande. Qual è il numero più grande che il VIC può gestire?
Per un piccolo programma utile a determinare sperimentalmente i numeri più grandi e più piccoli che il VIC può gestire, si veda il capitolo 6 sulla rappresentazione dei dati.
L'errore di *overflow* causa l'interruzione del programma. Per continuare si deve battere:

```
RUN
?
```

Per controllare l'effetto degli spazi all'inizio dei dati, battete:

```
      5
    5
  ?
```

Si provino varie combinazioni di spazi all'inizio, all'interno ed alla fine del testo, per capire come il VIC li gestisce.

```
?


?


?
SOLO 'RETURN'.

?
```

Battendo solo RETURN, viene utilizzato il valore precedente. Per terminare questo programma si premono simultaneamente i tasti RETURN e STOP. Occorrerà qualche tentativo per ottenere il BREAK IN 4000. Un altro modo per terminare questo programma è quello di premere simultaneamente i tasti STOP e RESTORE. Ciò pulisce lo schermo e provoca il ritorno ai colori standard.
Ricordatevi che se si scopre una combinazione che provoca un messaggio di errore e il READY, bisogna battere:

```
RUN
```

per continuare la vostra prova di ingresso dati.

Alcuni caratteri non sono permessi per l'INPUT con variabile in virgola mobile.

```
RUN
?
5 + 6
? REDO FROM START
?
```

Questo messaggio di errore indica che i dati di ingresso non sono accettabili. Ciò non termina il programma, ma concede un'altra possibilità. Cosa vi aspettate se battete:

```
BCD

?
```

Poiché l'istruzione INPUT si aspettava un numero in virgola mobile, una lettera (come la B) verrebbe considerata inaccettabile.
Provate con altri simboli per determinare ciò che è lecito.

```
?



?



?
```

Se non è stato provato qualcosa come:

```
123,456
```

provatelo ora.
La risposta:

```
? EXTRA IGNORED
 123
?
```

significa che sono stati forniti dati in eccesso alla richiesta dell'INPUT. La virgola è un separatore di dati.

'RETURN' 'STOP' *(contemporaneamente o* STOP *e* RESTORE)
OK

2. INPUT *di stringhe* Precedentemente si è cercato, senza successo, di inserire caratteri non numerici. L'errore era causato dal fatto che l'INPUT era del tipo in virgola mobile.
L'INPUT è utilizzato per inserire stringhe o valori numerici dalla tastiera. Funziona essenzialmente nello stesso modo sia in caso di numeri sia in caso di stringhe.

Per esempio:

```
NEW
10 INPUT A$,A
20 PRINT A$,A
RUN
?
```

A questo punto è possibile inserire una stringa di caratteri, una virgola ed un valore numerico. Il numero totale di caratteri in ingresso deve essere minore di 72. La virgola separa i due dati in ingresso.

```
ATEST=,10
ATEST=                  10
```

Nell'esempio precedente si può inserire la stringa desiderata e poi battere RETURN. A seguito di ciò, il sistema restituirà un ? ed attenderà l'inserimento del valore numerico. Ogni valore potrà essere introdotto su linee separate battendo RETURN. Per introdurre valori su di una stessa linea, occorre usare la virgola per separare i dati.

ESERCIZIO Utilizzate il precedente programma per inserire i dati uno alla volta.

```
RUN
?
ATEST=
??
10
ATEST=                  10
```

ESERCIZIO Programma per il controllo di validità per le stringhe di dati.

```
NEW
2000 INPUT X$:PRINTX$:GO TO 2000
RUN
?
ASDE
ASDE
?
```

I caratteri alfabetici sono leciti per variabili stringa X$, ma non per variabili numeriche in virgola mobile.

```
"RISPOSTA"
RISPOSTA
?
```

Si noti che le virgolette sono state eliminate, cioè non sono state incluse nella stringa.

```
BATTI""VIRGOLETTE
BATTI""VIRGOLETTE
?
```

Le virgolette vengono rimosse solo all'inizio di una linea.

```
    "AFR"
AFR
```

Gli spazi all'inizio e le virgolette sono stati rimossi.

```
   FGH
FGH
?
```
'RETURN' 'STOP' *(contemporaneamente o* STOP *e* RESTORE)
```
READY.
```

Gli spazi all'inizio sono stati rimossi. Modificate il programma di prova per vedere il trattamento degli spazi che sono alla fine della stringa.

```
2000 INPUTX$:Z$=X$+X$:PRINTZ$:GO TO 2000
RUN
?
XC
XCXC
?
```

Quando non vengono specificati gli spazi, l'assegnazione Z$ = X$ + X$ concatena la stringa di ingresso con se stessa.

```
XC
XCXC
?
```

Gli spazi alla fine della stringa non sono stati considerati.

```
     ABC
  ABC   ABC
  ?
```

Tutti gli spazi all'inizio e alla fine della stringa sono stati rimossi.

```
  XYZ  AB
XYZ  AB  XYZ  AB
?
```

Gli spazi all'interno sono permessi.

```
"   ABC    "
   ABC         ABC
?
```

Tutti gli spazi all'interno delle virgolette sono stati inclusi.

```
" A
 A A
?
```

Le virgolette all'inizio conserveranno gli spazi che le seguono.

```
A "
A "A "
?
"A
AA
?
ABC,
?EXTRA IGNORED
ABCABC
?
```

La virgola è considerata un separatore di dati e tratta le variabili stringa come se fossero variabili numeriche.

```
"ABC,
ABC,ABC,
?
```

Le virgolette permettono alla virgola di essere parte della stringa e non invece un separatore.

Si provino vari esempi utilizzando caratteri speciali entro una stringa con e senza virgolette.

?

?

'RETURN' 'STOP'

Molti programmi usano una codifica numerica per indicare delle scelte. Mediante le stringhe, ad esempio, non è più necessario indicare il SI con un 1, oppure il NO con uno 0. Si può rendere più semplice l'uso del programma ad utenti con poca esperienza facendo in modo che accetti parole in ingresso.

ESERCIZIO Trovate un algoritmo che consenta una scelta nell'ambito di un programma, mediante un INPUT di tipo stringa.

```
NEW
1500 INPUT"SI, NO O FORSE";Z$
1510 IF Z$="SI" THEN PRINT "VA BENE"
1520 IF Z$="NO" THEN PRINT "PAZIENZA"
1530 IF Z$="FORSE" THEN PRINT "DECIDITI!"
1540 IF Z$="" THEN STOP
1550 GO TO 1500
RUN
SI, NO O FORSE?
```

Assicuratevi di usare il punto e virgola tra le ultime virgolette ed il nome della variabile Z$ nella linea 1500. Ricordate che la stringa di messaggio deve essere contenuta in una linea (23 caratteri).
Se appare un messaggio di errore, ricontrollate la linea indicata nel messaggio stesso mediante:

LIST numero di linea

Gli errori più comuni sono un numero sbagliato di virgolette, errori di sintassi di parole chiave, un $ omesso od una virgola al posto del punto e virgola.
Fate i seguenti esperimenti sul programma.

```
SI

SI, NO O FORSE?
NO

SI, NO O FORSE?
FORSE

SI, NO O FORSE?
RISPOSTA NON VALIDA

SI, NO O FORSE?
'RETURN'
```

5.3 READ, DATA E RESTORE

L'istruzione READ viene utilizzata per permettere al programmatore di cambiare facilmente il valore di variabili (parametri) nel programma. Il READ necessita di una istruzione DATA nel programma.
I dati possono essere inseriti in un programma mediante l'istruzione DATA e la corrispondente istruzione READ. Un'istruzione DATA deve avere un valore corrispondente per ciascuna variabile READ. Tuttavia, RESTORE reinizializza l'istruzione DATA, che così potrà essere letta più volte.

ESERCIZIO Usate le istruzioni DATA, READ e RESTORE per inserire stringhe nel vostro programma.

```
NEW
10 DATA "A","B"
20 DATA C,D
30 READ A$,B$
40 PRINT A$,B$
50 READ C$,D$
60 PRINT C$,D$
70 RESTORE
80 GOTO 30
```

```
RUN
A               B
C               D
A               B
C               D
 .
 .
 .
 .
```

Il programma ora si trova in un ciclo infinito perché RESTORE permette di rileggere le istruzioni DATA più volte.

```
'STOP'
BREAK IN 40
70
RUN
A               B
C               D
?OUT OF DATA
 ERROR IN 30
```

Quando viene rimosso RESTORE (linea 70), le istruzioni DATA vengono lette una sola volta. Si noti che in un'istruzione DATA una stringa di caratteri non necessita delle virgolette. Vedere linea 20.

CAPITOLO SEI

Rappresentazione dei dati

6.1 IL NUMERO PIÙ GRANDE

Per ottenere il più grande numero in virgola mobile che il VIC può gestire, eseguite il seguente programma.

ESERCIZIO Ricerca del massimo numero in virgola mobile.

```
NEW
200 I=1
210 I=I+I
220 PRINT I
230 GO TO 210
RUN
```

Verrà visualizzata una lunga serie di numeri, che iniziano con:

```
2
4
8
16
```

l'ultimo numero sarà:

```
4.25352959E+37
8.50705917E+37
```

seguito da:

```
?OVERFLOW
 ERROR IN 210
```

Questo significa che il numero 1.701411835E+38 era troppo grande per essere visualizzato.

```
?1.701411834E+38
 1.701411834E+38
?1.701412E+38
?OVERFLOW
 ERROR
```

Da ciò si nota che 1.701411834E+38 è il numero più grande che il VIC può gestire.

ESERCIZIO Modificate il precedente programma per visualizzare il più grande numero negativo in virgola mobile che il VIC può gestire. Se il segno del risultato si alterna tra il più ed il meno, bisogna provare con una diversa modifica. Il risultato dovrebbe dimostrare che il massimo numero in virgola mobile gestibile dal VIC è indipendente dal segno.

____

____

____

____

Il più grande numero negativo è ________.

6.2 IL NUMERO PIÙ PICCOLO

Per ottenere il più piccolo numero in virgola mobile che il VIC può gestire, eseguite il seguente programma.

ESERCIZIO Ricerca del minimo numero in virgola mobile.

```
NEW
100 I=1
110 I=I/2
120 PRINT I
130 IF I<>0 THEN 110
RUN
```

Verrà visualizzata una lunga serie di numeri, che iniziano con:

```
.5
.25
.125
.
.
.
```

Gli ultimi due numeri saranno:

```
2.93873588E-39
0
```

Questo significa che il numero 1.46936794E-39 era troppo piccolo per essere visualizzato con un numero diverso da zero.

```
?2.93873588E-39
 2.93873588E-39
?2.93873587E-39
 0
```

Da ciò si nota che 2.93873587E−39 è il numero più piccolo che il VIC può gestire. Qualsiasi numero più piccolo verrà approssimato a zero.

ESERCIZIO Modificate il precedente programma per visualizzare il più piccolo numero negativo in virgola mobile che il VIC può gestire.

____

____

____

____

Il più piccolo numero negativo che il VIC può rappresentare è ________.

6.3 CAMPO DI VARIAZIONE PER I NUMERI INTERI

Per trovare il più grande numero intero che il VIC può gestire:

```
NEW
300 I%=0
305 I%=I%+1
310 J%=2*I%-1
320 PRINT J%
330 GO TO 305
RUN
```

Verrà visualizzata una serie di numeri:

```
1
3
7
.
.
.
32767
?ILLEGAL QUANTITY
 ERROR IN 310
```

Questo significa che il numero 65535 è troppo grande per essere visualizzato come numero intero. Provate a battere

```
I%=32768
```

Da ciò si nota che 32767 è il numero più grande che può essere rappresentato come intero.

ESERCIZIO Modificate il precedente programma per visualizzare il più grande intero negativo.

____

____

____

____

6.4 SPAZIO DI MEMORIA UTILIZZATO

Per determinare lo spazio di memoria richiesto per l'immagazzinamento dei dati, si utilizza la funzione FRE, che restituisce la quantità di memoria ancora disponibile.
Queste sono le risposte per una macchina a 5 K. Se si dispone di maggior memoria, si prenda nota del valore.

ESERCIZIO Determinate la quantità di spazio utilizzato da un singolo valore in virgola mobile.
Innanzitutto, determinate la quantità di memoria rimanente dopo aver riservato 100 locazioni in virgola mobile.

```
NEW
DIM A(100)
? FRE(0)
3069
```

Poi, determinate la quantità di memoria rimanente dopo aver riservato 101 locazioni in virgola mobile.

```
NEW
DIM A(101)
? FRE(0)
3064
```

La differenza (5) rappresenta il numero di byte richiesti per un numero in virgola mobile. Ora determinate quanto spazio occupa un valore intero.

```
NEW
DIM A%(100)
? FRE(0)
3372
NEW
DIM A%(101)
? FRE(0)
3370
```

La differenza (2) rappresenta il numero di byte richiesti per un numero intero.

ESERCIZIO Determinate la quantità di spazio utilizzato da una singola stringa.

```
NEW
DIM A$(100)
? FRE(0)
3271
NEW
DIM A$(101)
? FRE(0)
3268
```

Occorrono 3 byte in più per ogni ulteriore variabile stringa. Tuttavia ogni stringa può contenere fino a 255 caratteri.
Oltre allo spazio utilizzato per ciascun valore, si devono aggiungere 7 byte per il nome della variabile e per i suoi puntatori.

6.5 NUMERO DI CIFRE SIGNIFICATIVE

Il VIC può visualizzare 6 cifre in un numero anche se può contenere più informazione in memoria.

ESERCIZIO Determinate il numero di cifre visualizzabili per numeri in virgola mobile.

```
PRINT 123456789
123456789
```

Questo indica che possono essere visualizzate 9 cifre.

```
PRINT 1234567890
1.23456789E+09
```

Ciò dimostra che la decima cifra non è visualizzabile.

ESERCIZIO Trovate il numero massimo di cifre significative memorizzabili.

```
A=1234567890: PRINT A
1.23456789E+09
B=1234567891: PRINT B
1.23456789E+09
```

Anche se i numeri inseriti differivano di uno nella decima cifra, il video mostra che sono uguali. Lo sono realmente?

ESERCIZIO Controllate se i numeri inseriti sono davvero uguali.

```
IF A<>B THEN PRINT "<>"
<>
```

La risposta dimostra che il VIC può distinguere la differenza di 1 nella decima cifra anche se non la visualizza. Può il VIC distinguere una differenza di 1 nella undicesima cifra?

```
C=12345678900: PRINT C
1.23456789E+10
D=12345678901: PRINT D
1.23456789E+10
```

I valori visualizzati sembrano essere uguali. Lo sono? Provate

```
IF C<>D THEN PRINT "<>"
```

Sembra evidente che il VIC non può distinguere tra due numeri che differiscono di 1 nell'undicesima cifra.
Il numero di cifre significative visualizzabile è 9, mentre il numero di cifre in memoria è 10.

6.6 ARROTONDAMENTO

Siccome il VIC può gestire circa 10 cifre in memoria, mentre ne visualizza solo 9, viene effettuato un arrotondamento per la visualizzazione.

ESERCIZIO Determinate come il VIC arrotonda i numeri per la visualizzazione.

```
NEW
100 A=1234567890
110 FOR I=1 TO 15
120 B=A+I
130 PRINT "A+";I;"=";B
140 NEXT I
RUN
A+ 1 = 1.23456789E+09
.
.
.
```

```
A+ 5 = 1.2345679E+09
.
.
.
A+ 15 = 1.23456791E+10
```

Da questa visualizzazione si vede che 1234567895 è stato arrotondato a 1.2345679E+09 e che 1234567905 è stato arrotondato a 1.23456791E+09. Il VIC arrotonda per eccesso la nona cifra quando la decima è maggiore o tutt'al più uguale a 5, e per difetto in caso contrario.

CAPITOLO SETTE

Il registratore a cassette

Il VIC da solo non ha la possibilità di salvare programmi o dati. Ogni volta che viene spento, tutte le informazioni vengono perdute e dovranno essere reinserite alla successiva accensione. Per questa ragione, uno dei primi acquisti sarà probabilmente un supporto per la memorizzazione dei dati. Si hanno tre scelte: un registratore a cassette, un floppy disk o ambedue. Poiché il registratore è notevolmente meno costoso e più facilmente reperibile, l'immagazzinamento dei programmi sarà descritto con riferimento a questa unità. Se siete tanto fortunati da possedere un floppy disk, è sufficiente aggiungere ,8 alla fine di ogni comando.

ESEMPIO

```
SAVE "NOMEFILE"   SAVE "NOMEFILE",8
VERIFY "NOMEFILE" VERIFY "NOMEFILE",8
LOAD "NOMEFILE"   LOAD "NOMEFILE",8
```

La risposta ottenuta utilizzando il floppy disk sarà leggermente diversa da quella del registratore, ma il risultato finale sarà uguale.
Ed ora possiamo cominciare.

7.1 SALVARE UN PROGRAMMA (SAVE)

Poiché la battitura di un programma è un lavoro lungo, è preferibile memorizzarlo. Il comando che esegue ciò è il SAVE "nome del programma".

ESERCIZIO Registrate un programma su cassetta. Inserite una cassetta (che non contenga altre informazioni) nel registratore.

```
10 REM PROVA PER SAVE
SAVE
PRESS RECORD & PLAY ON
TAPE
OK (quando avete premuto PLAY e REC sul registratore)
SAVING
READY.
```

Se lo desiderate, è possibile etichettare con un nome il programma battendo SAVE"nome del programma". Il nome può essere lungo fino ad un massimo di 83 caratteri.

ESERCIZIO Registrate un programma su cassetta con il nome "ABC". Riavvolgete la cassetta.

```
SAVE"ABC"
PRESS RECORD & PLAY ON
TAPE
OK (quando avete premuto PLAY e REC sul registratore)
WRITING ABC
READY.
```

Attenzione! Il sistema non esegue il controllo di quale tasto del registratore viene premuto. Qualsiasi tasto venga premuto, il sistema assume che sia quello giusto.

7.2 VERIFICA DI UN PROGRAMMA (VERIFY)

Per essere sicuri di aver registrato correttamente il programma sulla cassetta, lo si può confrontare con quello in memoria mediante il comando VERIFY. Innanzitutto riavvolgete la cassetta.

ESERCIZIO Verificate che l'informazione appena registrata è corretta.

```
VERIFY
PRESS PLAY ON TAPE
OK
SEARCHING
FOUND
VERIFYING
OK
READY.
```

Se si verifica un errore di confronto, si ottiene ?VERIFY ERROR. Se questo succede, riavvolgete il nastro e riprovate. Poiché la lettura e la registrazione su cassetta non sono molto affidabili, riprovando è possibile risolvere il problema.
Se si batte VERIFY senza il nome, il sistema cercherà di confrontare il primo programma che trova sulla cassetta con quello in memoria. Se è stato battuto VERIFY e il nome del programma, il sistema cercherà di trovare il programma relativo. Verrà visualizzato il nome di ogni programma incontrato.
La cassetta potrà essere tolta dal registratore senza riavvolgere il nastro; quindi, in un secondo tempo, potrà essere reinserita registrando da quel punto un nuovo programma. Il comando SAVE, prima di eseguire la registrazione del programma, avvolge per 10-15 secondi il nastro della cassetta.

ESERCIZIO Salvate un programma e dimostrate che VERIFY coglie le eventuali differenze.
Per prima cosa riavvolgete una cassetta vergine.

```
NEW
10 PRINT "PROVA IL VERIFY"
20 INPUT L,W
30 A=L*W
40 PRINT A,L,W
50 END
SAVE "A"
PRESS RECORD & PLAY ON
TAPE
OK
SAVING
READY.
```

Ciò inserisce un piccolo programma in memoria e lo salva su cassetta con il nome A. Ora riavvolgete la cassetta.

```
VERIFY
PRESS PLAY ON TAPE
OK
SEARCHING
FOUND A
VERIFYING
OK
READY.
```

Il programma è stato registrato correttamente. Non riavvolgete la cassetta. È sufficiente premere lo STOP sul registratore.

```
SAVE"ABC"
PRESS RECORD & PLAY ON
TAPE
OK
SAVING ABC
RADY.
```

Questa operazione ha salvato una seconda copia del programma sulla cassetta. La seconda copia segue la prima sul nastro ed è identificata dal nome ABC.
Riavvolgete la cassetta.
Per ottenere un errore di verifica, cambieremo il programma in memoria.

```
10 PRINT "INTRODUCE UN ERRORE"
VERIFY "ABC"
PRESS PLAY ON TAPE
OK
SEARCHING FOR ABC
FOUND A
FOUND ABC
VERIFYING
?VERIFY
 ERROR
READY.
```

7.3 CARICARE UN PROGRAMMA (LOAD)

Prima di caricare un programma, controlliamo ciò che già risiede in memoria.

```
LIST 10
10 PRINT "INTRODUCE UN ERRORE"
```

Per ricaricare il programma originale, riavvolgete il nastro e battete:

```
LOAD
PRESS PLAY ON TAPE
OK
```

Il comando LOAD cancella automaticamente il contenuto della memoria e vi inserisce il primo programma della cassetta. Dopo aver premuto il tasto PLAY sul registratore verranno visualizzate le seguenti risposte.

```
SEARCHING
FOUND A
LOADING
READY.
LIST 10
10 PRINT "PROVA IL VERIFY"
```

Se si vuol caricare il programma con il nome ABC, riavvolgete il nastro e battete LOAD"ABC".

ESERCIZIO Caricate un programma precedentemente salvato.

```
LOAD "ABC"
PRESS PLAY ON TAPE
SEARCHING FOR ABC
FOUND A
FOUND ABC
LOADING ABC
READY.
```

Se sulla cassetta risiede più di un programma e si batte LOAD, verrà caricato il primo programma trovato. Per esempio, se si hanno tre programmi su di una cassetta ed il nastro è posizionato alla fine del primo programma, il LOAD caricherà il secondo programma.
Se si batte LOAD"ABC", viene effettuata una ricerca del programma ABC. Appena trovato, esso verrà caricato. Durante la ricerca verrà visualizzato un elenco di tutti i programmi precedenti ABC. Per esempio, si supponga che vi siano quattro programmi sulla cassetta, i primi due con nome 1 e $ rispettivamente, il terzo senza nome ed il quarto con nome VIC.
Se viene dato il comando LOAD"VIC", il sistema risponde come segue:

```
LOAD "VIC"
PRESS PLAY ON TAPE
OK
SEARCHING FOR VIC
FOUND 1
FOUND $
FOUND
FOUND VIC
LOADING
READY.
```

Se non si sa cosa contiene la cassetta, basta battere il LOAD e il nome di un programma non registrato verrà così visualizzato ogni programma contenuto nella cassetta.

Attenzione! Caricare un programma da cassetta non è una operazione sempre sicura. L'errore più comune è che il sistema trova il programma con il nome richiesto, ma non riesce a caricarlo. L'unico modo di ovviare questo inconveniente è di riavvolgere il nastro e ritentare. Spesso, al secondo o terzo tentativo si avrà successo.

CAPITOLO OTTO

Salti condizionati ed incondizionati

Il Basic del VIC ha due istruzioni per il salto. L'istruzione GO TO (oppure GOTO) è un salto incondizionato ad una ben definita linea di programma. L'istruzione ON X GOTO è, invece, un salto incondizionato ad una linea scelta tra più linee. La linea prescelta è specificata dal valore di X e dall'insieme di numeri di linea indicati. In certi casi, lo spazio tra GO e TO causa un errore di sintassi. Perciò utilizzate sempre GOTO.

8.1 GOTO

Un salto incondizionato (cioè un trasferimento del controllo) ad una singola linea viene indicato con GOTO nnn, ove nnn è un numero di linea. Il numero di linea deve essere una costante numerica. Non sono permesse le variabili e le costanti stringa.

ESERCIZIO Controllo dei numeri di linea leciti nell'istruzione GOTO.

```
NEW
100 PRINT"GOTO 100":END
GOTO 100
GOTO 100
```

```
GOTO "100"
?UNDEF'D STATEMENT
 ERROR

S=100:GOTO S
?UNDEF'D STATEMENT
 ERROR

S$="100":GOTO S$
?UNDEF'D STATEMENT
 ERROR
```

È possibile vedere che una costante numerica è l'unico modo lecito per specificare la destinazione del GOTO.
Il numero di linea al quale viene trasferito il controllo mediante GOTO deve essere presente, altrimenti verrà dato un messaggio di errore.

```
GOTO 110
?UNDEF'D STATEMENT
 ERROR
```

Anche se in questi esempi è accettabile GO TO, è sempre consigliabile battere GOTO, poiché in certi casi lo spazio non è permesso.

8.2 ON X GOTO

Il salto a più linee è indicato con ON X GOTO n1,n2,n3,...,nn ove X è la variabile che seleziona a quale linea saltare tra le linee indicate dai numeri n1,n2,n3,...,nn. La destinazione è l'X-esimo numero. Per esempio, se X è uguale a 2, allora il controllo è trasferito alla seconda linea (n2) tra quelle indicate nella lista. Se X è uguale a 4, allora il controllo è trasferito alla quarta linea (n4) tra quelle indicate nella lista.

ESERCIZIO Controllate come funziona l'istruzione ON X GOTO.

```
1 PRINT "RIGA N.1":STOP
2 PRINT "RIGA N.2":STOP
3 PRINT "RIGA N.3":STOP
4 PRINT "RIGA N.4":STOP
5 PRINT "RIGA N.5":STOP
100 X=2
110 ON X GOTO 1,2,3,5
```

```
120 PRINT "SALTATO DAL GOTO"
RUN 100
RIGA N.2
BREAK IN 2
```

Se ottenete RIGA N.1 ed il BREAK IN 1, è stato probabilmente battuto semplicemente RUN e non RUN 100. Provate con RUN 100. Se si ottiene un ?SINTAX ERROR IN 110, occorre eliminare lo spazio tra il GO ed il TO.

```
100 X=4
RUN 100
RIGA N.5
BREAK IN 5
```

Si noti che ON X trasferisce il controllo alla linea indicata dal quarto numero della lista e non alla linea 4.

ESERCIZIO Dimostrate come ON X si comporta con valori esterni all'intervallo consentito.

```
100 X=0
RUN 100
SALTATO IL GOTO

100 X=5
RUN 100
SALTATO IL GOTO
```

Se la variabile ha un valore diverso dalla quantità di numeri di linea a disposizione, viene eseguita la successiva istruzione.

ESERCIZIO Dimostrate l'intervallo lecito per i valori della variabile X in ON X...

```
100 X=255
RUN 100
SALTATO IL GOTO

100 X=256
RUN 100
?ILLEGAL QUANTITY
 ERROR IN 110
```

```
100 X=-1
RUN 100
?ILLEGAL QUANTITY
 ERROR IN 110
```

Questo dimostra che il valore di X deve appartenere all'intervallo 0-255.

ESERCIZIO Utilizzate ON X GOTO per selezionare una tra quattro possibilità.
Si supponga di voler scrivere un programma che serva a gestire un archivio di indirizzi e che esistano quattro possibilità.

1. Aggiungere un nuovo nome alla lista.
2. Modificare un nome già esistente.
3. Visualizzare la lista.
4. Contare il numero di nomi presenti nella lista.

È possibile studiare il programma in modo che accetti all'ingresso i numeri dall'1 al 4 per selezionare le possibilità.

```
NEW
100 PRINT "MENU MAILING LIST"
110 INPUT "COSA VUOI FARE? ";A
130 ON A GOTO 1000,2000,3000,4000
```

La scelta dei numeri di linea è alquanto arbitraria. Si possono rendere i programmi più leggibili sviluppando un modello coerente (stile). È facile vedere che per A = n, il controllo si trasferisce alla linea n000.

```
150 GOTO 110
1000 PRINT "AGGIUNTA DI UN NOME"
1999 GOTO 150
2000 PRINT "CAMBIAMENTO DI UN NOME"
2990 GOTO 150
3000 PRINT "LISTA DEI NOMI"
3990 GOTO 150
4000 PRINT "CONTO DEI NOMI"
4990 GOTO 150
```

Per ogni tipo di scelta si può inserire un commento esplicativo. Questo commento fornisce un'indicazione della funzionalità o meno del programma dopo una determinata scelta. Successivamente, le istruzioni PRINT possono essere sostituite dai passi di programma per compiere il lavoro richiesto.

Può sembrare strano avere la linea 150 (GOTO 110) e che il commento termini con un GOTO 150. Un GOTO ad un altro GOTO sembrerebbe molto inefficiente. Infatti lo è. Un GOTO diretto è più veloce. Tuttavia, se si utilizza questo metodo di ritornare sempre alla linea dopo un'istruzione ON X GOTO, non sarà necessario indovinare dove si trasferirà il controllo del programma. Ciò facilita la ricerca degli errori (*debugging*). Ora si provino tutti gli input leciti.

```
RUN
MENU MAILING LIST
COSA SCEGLI?

1
AGGIUNTA DI UN NOME
COSA SCEGLI?

2
CAMBIAMENTO DI UN NOME
COSA SCEGLI?

3
LISTA DEI NOMI
COSA SCEGLI?

4
CONTO DEI NOMI
COSA SCEGLI?
```

Per terminare questo programma, premete simultaneamente RETURN e STOP. Il programma deve essere registrato su cassetta con il titolo POSTA1. Se vi siete dimenticati come si registra un programma su cassetta, ripassate il capitolo 7. Se non potete registrare il programma, occorrerà reinserirlo. Più avanti si utilizzerà POSTA1 aggiungendo altre caratteristiche.

ESERCIZIO Controllate come POSTA1 reagisce ad errori in ingresso. Se il VIC è stato spento, ricaricate il programma. Per ricaricarlo battete:

```
LOAD
PRESS PLAY ON TAPE
OK
SEARCHING
FOUND POSTA1
LOADING
```

Ora si può controllare il programma.

```
RUN
MENU MAILING LIST
COSA SCEGLI?
0
COSA SCEGLI?
5
COSA SCEGLI?
```

Il programma non gestisce bene gli input errati.
È possibile approfittare del fatto che un numero esterno all'intervallo provoca il passaggio alla successiva linea di programma per verificare un eventuale errore. Per terminare il programma basta premere RETURN e STOP oppure STOP e RESTORE.

```
140 PRINT "DA 1 A 4"

RUN
MENU MAILING LIST
COSA SCEGLI?
0
DA 1 A 4
COSA SCEGLI?
```

Salvate questa versione del programma con il nome POSTA2. Più tardi lo utilizzeremo come base per sviluppare un programma riferito ad un utente generico. Poiché il VIC può gestire sia stringhe di caratteri che numeri, non vi è motivo per cui POSTA non funzioni correttamente anche con scelte indicate da caratteri, anziché solo numeri. Nel capitolo sull'istruzione IF, verrà spiegato un ulteriore miglioramento per gestire numeri negativi anche grandi, come valore per la A.

CAPITOLO NOVE

L'istruzione IF

L'istruzione IF permette ad un programma di prendere delle decisioni, basandosi, per la scelta, sui parametri di ingresso o sul valore di una variabile.

9.1 LA FORMA THEN

La forma più generale dell'istruzione IF è: IF espressione THEN istruzione. Quando l'espressione è vera (quando, cioè, il suo valore è diverso da zero), viene eseguita l'istruzione. Quando l'espressione è falsa (quando, cioè, il suo valore è uguale a zero), l'istruzione è ignorata e viene eseguita la successiva linea di programma.

ESERCIZIO Utilizzate la forma THEN dell'istruzione IF.

```
NEW
100 A=1
110 IF A=1 THEN PRINT "A=1"
120 PRINT "LINEA 120"

RUN
A=1
LINEA 120
```

```
100 A=2
RUN
LINEA 120
```

9.2 LA FORMA GOTO

L'istruzione IF non necessita sempre del THEN. È infatti lecita la forma: IF espressione GOTO nnn. Tale forma è del tutto equivalente alla forma: IF espressione THEN GOTO nnn. Un terzo modo di specificare la stessa operazione è: IF espressione THEN nnn.
Tutte e tre le forme danno lo stesso risultato. Se l'espressione è vera, l'esecuzione continua dalla linea nnn. Se, invece, è falsa, l'esecuzione continua con la successiva linea di programma.

ESERCIZIO Provate la forma GOTO dell'istruzione IF. Utilizzate parte del programma già inserito.

```
130 STOP
140 PRINT "A=1"
150 STOP
110 IF A=1 GOTO 140
RUN
LINEA 120
BREAK IN 130
```

Poiché A=2, la prova IF è negativa. Questo causa l'esecuzione delle linee 120 e 130.

```
100 A=1
RUN
A=1
BREAK IN 150
```

In questo caso A=1 e vengono eseguite le linee 140 e 150.

ESERCIZIO Usate la forma breve del GOTO nell'istruzione IF.

```
110 IF A=1 THEN 140
RUN
A=1
BREAK IN 150
```

```
100 A=2
RUN
LINEA 120
BREAK IN 130
```

Queste due prove mostrano gli stessi risultati di quelli dell'esercizio precedente.

9.3 LE ISTRUZIONI MULTIPLE

La forma THEN dell'istruzione IF può avere istruzioni multiple dopo il THEN. Le istruzioni multiple vengono eseguite solo se la verifica è positiva.

ESERCIZIO Dimostrate che le istruzioni multiple dopo il THEN vengono eseguite.

```
NEW
100 A=1
110 IF A=1 THEN PRINT"A=1":A=2:PRINT"ADESSO A =";A
120 PRINT "LINEA 120"
130 STOP
RUN
A=1
ADESSO A = 2
LINEA 120
BREAK IN 130
```

Quando la prova IF è positiva, vengono eseguite tutte le istruzioni dopo il THEN.

```
110 IF A=1 THEN PRINT"A=1":GOTO 130: PRINT"MAI QUI!"
RUN
A=1
BREAK IN 130
```

L'istruzione dopo GOTO 130 non potrà mai essere eseguita. Infatti, il GOTO impedisce l'esecuzione in sequenza dell'istruzione PRINT"MAI QUI!". Poiché quel PRINT non ha un numero di linea, non può essere eseguito da un altro GOTO. Essendo permesso un solo numero di linea sulla stessa riga, non è possibile inserire un numero prima di PRINT.

ESERCIZIO Dimostrate che non è possibile inserire un secondo numero di linea sulla stessa riga.

```
110 IF A=1 THEN PRINT"A=1":GOTO 130:115 PRINT"MAI QUI!"
RUN
A=1
BREAK IN 130

GOTO 115
?UNDEF'D STATEMENT
 ERROR
```

Questo dimostra che il numero 115 non corrisponde ad una linea di programma. Solo una linea regolarmente memorizzata può essere eseguita. Eliminiamo GOTO 130 per vedere cosa succede con due numeri su di una stessa riga.

```
110 IF A=1 THEN PRINT"A=1":115 PRINT"MAI QUI!"
RUN
A=1
?SYNTAX
 ERROR IN 110
```

È possibile utilizzare ciò che è noto sull'istruzione IF per migliorare il programma POSTA2.

ESERCIZIO Aggiungete al programma POSTA2 un controllo di errore. Ricaricate il programma. Per rivedere come il programma gestisce dati errati in ingresso, provate:

```
RUN
MENU MAILING LIST
COSA SCEGLI?
-1
?ILLEGAL QUANTITY
 ERROR IN 130

RUN
MENU MAILING LIST
COSA SCEGLI?
256
?ILLEGAL QUANTITY
 ERROR IN 130
```

Le seguenti due linee controlleranno la validità dei dati inseriti, permettendo al programma di continuare.

```
120 IF A<0 THEN 140
122 IF A>255 THEN 140
RUN
MENU MAILING LIST
COSA SCEGLI?
-5
DA 1 A 4
COSA SCEGLI?
256
DA 1 A 4
COSA SCEGLI?
'RETURN' 'STOP' O 'STOP' 'RESTORE'
```

I risultati mostrano che il programma non si blocca più in caso di errore. Salvate il programma con il nome POSTA3.

ESERCIZIO Rendete il programma POSTA3 più semplice da eseguire per un utente non programmatore. Fate in modo che vengano accettate in ingresso le stringhe AGGIUNGERE, MODIFICARE, VISUALIZZARE e CONTARE al posto dei numeri 1, 2, 3 e 4. Caricate il programma POSTA3. Un abbozzo di programma per la gestione degli indirizzi del tipo ad ingresso alfabetico si trova in appendice A.

```
101 AT$(1)="AGGIUNGERE"
102 AT$(2)="MODIFICARE"
103 AT$(3)="VISUALIZZARE"
104 AT$(4)="CONTARE"
```

Ciò inizializza il vettore AT$ con le quattro parole chiave. Si può inserire qualsiasi stringa desiderata.

```
110 INPUT"COSA VUOI FARE";A$
112 IF A$="" THEN STOP
115 FOR A=1 TO 4
120 IF A$<>AT$(A) THEN 135
130 ON A GOTO 1000,2000,3000,4000
135 NEXT A
```

Confronta la stringa in ingresso con le parole chiave.

```
140 PRINT "INPUT SBAGLIATO!"
```

Cambia il messaggio per indicare l'errore.

```
RUN
MENU MAILING LIST
COSA VUOI FARE?
AGGIUNTA
AGGIUNTA DI UN NOME
COSA VUOI FARE?
```

Provate con stringhe lecite e non, per convincervi che il programma funziona. Per quale motivo non occorre controllare se A appartiene o meno all'intervallo 0-255? Salvate questo programma con il nome POSTA4.

CAPITOLO DIECI

Le subroutine

Quando si acquisisce maggiore pratica nella programmazione è inevitabile che i programmi diventino più lunghi e complessi. Questi programmi vengono eseguiti con più facilità e più in fretta mediante l'utilizzazione delle subroutine. Le subroutine servono ad evitare ridondanze nei programmi che utilizzano più volte le stesse istruzioni, semplificando da un lato l'organizzazione di un programma e la sua leggibilità, e dall'altro lato aiutando il programmatore nella correzione di eventuali errori.
Una volta che la subroutine sia stata corretta dagli errori, potrà essere utilizzata come base per l'elaborazione di altri programmi. Ciò è di particolare utilità se tale base non interferisce con altre basi.
Per utilizzare una subroutine bisogna ricordare due principi. Il primo è quello del collegamento (*linkage*). Quando si vuole usare la subroutine, essa deve essere chiamata dal programma. Quando la subroutine è completata, il controllo del programma deve tornare al punto in cui è stata chiamata. La fig. 6 illustra questo collegamento. Le subroutine sono di maggiore utilità se possono essere chiamate da diversi punti del programma. Per questa ragione è importante inserire nella subroutine una funzione richiesta frequentemente ed in punti diversi nel programma.
Il secondo principio è quello del passaggio dei parametri. Poiché il Basic non ha un sistema per passare i parametri con la chiamata di una subroutine, sarà necessario stabilire delle convenzioni. I *parametri* sono i dati sui quali la subroutine deve lavorare. I parametri (detti anche argomenti) possono essere inseriti (nella subroutine) e restituiti (valori di ritorno dalla subroutine nel programma principale).
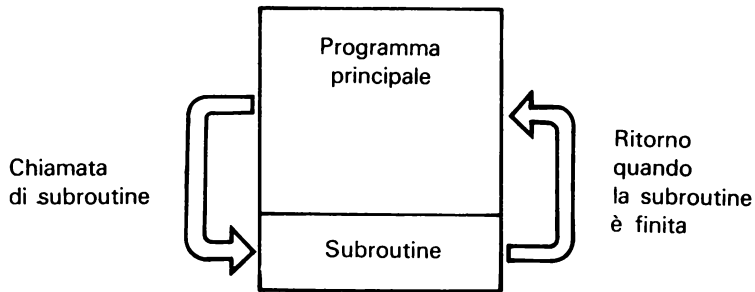
Fig. 6 Collegamento di subroutine.

10.1 GOSUB

Per chiamare una subroutine, deve essere eseguita l'istruzione GOSUB nnn. Il controllo passa immediatamente alla linea nnn. Il GOSUB funziona in maniera analoga al GOTO, con l'eccezione che il GOSUB, dopo essere stato eseguito, restituisce il controllo alla linea chiamante.

10.2 RETURN

L'istruzione RETURN permette di terminare una subroutine e di restituire il controllo là dove la subroutine è stata chiamata.

ESERCIZIO Dimostrate come vengono utilizzate le subroutine.

```
NEW
100 PRINT "PROGRAMMA PRINCIPALE"
110 GOSUB 1000
120 PRINT "PROGRAMMA PRINCIPALE  DOPO LA SUBROUTINE"
130 STOP
1000 PRINT "SUBROUTINE 1000"
1190 RETURN

RUN
PROGRAMMA PRINCIPALE
SUBROUTINE 1000
PROGRAMMA PRINCIPALE
DOPO LA SUBROUTINE
BREAK IN 130
```

ESERCIZIO Dimostrate ciò che succede se viene eseguito RETURN senza GOSUB.

```
RUN 1000
SUBROUTINE 1000
?RETURN WITHOUT GOSUB
 ERROR IN 1190

110 GOTO 1000
RUN
PROGRAMMA PRINCIPALE
SUBROUTINE 1000
?RETURN WITHOUT GOSUB
 ERROR IN 1190
```

ESERCIZIO Controllate la dimensione massima di un "nido" di subroutine. Cioè, quante GOSUB possono essere eseguite senza un RETURN?

```
NEW
5 I=-1
10 I=I+1
20 GOSUB 10
RUN
?OUT OF MEMORY
 ERROR IN 10
PRINT I
23
```

La dimensione massima di un nido di subroutine permessa dal VIC è 23.

10.3 ON X GOSUB

L'istruzione GOSUB è simile all'istruzione GOTO in quanto viene specificata una singola destinazione (numero di linea). Così come si possono fare salti multipli incondizionati con ON X GOTO n1,n2,...,nn, è possibile selezionare una tra più subroutine con ON X GOSUB n1,n2,...,nn.

ESERCIZIO Mostrate come funziona ON X GOSUB.

```
NEW
100 INPUT "1,2 O 3";I
110 ON I GOSUB 1000,2000,3000
120 PRINT "PROG. PRINCIPALE"
```

```
130 GOTO 100
1000 PRINT "PRIMA SUBROUTINE"
1990 RETURN
2000 PRINT "SECONDA SUBROUTINE"
2990 RETURN
3000 PRINT "TERZA SUBROUTINE"
3990 RETURN

RUN
1,2 O 3?
1
PRIMA SUBROUTINE
PROG. PRINCIPALE
1,2 O 3?

0
PROG. PRINCIPALE
1,2 O 3?
```

Quando I vale 0 oppure è maggiore di 3, l'istruzione ON X GOSUB si comporta nello stesso modo di ON X GOTO, cioè viene eseguita la successiva linea di programma. Per terminare il programma:

```
1E40
?OVERFLOW
 ERROR IN 100
```

CAPITOLO UNDICI

Le stringhe

Ogni computer può riconoscere un certo numero di caratteri o simboli. Questi possono essere cifre, lettere alfabetiche o caratteri speciali. Nel nostro caso vengono presi in considerazione i simboli sulla tastiera del VIC. Ad ogni carattere (simbolo) è assegnato un numero che rappresenta il carattere stesso all'interno del computer. Si definisce *stringa* un insieme di uno o più caratteri. Essa può essere un messaggio, un nome, un indirizzo, ecc.
Per consentire la comunicazione tra i computer, stabilendo una normativa in merito, si utilizza il codice ASCII (*American Standard Code for Information Interchange*) per i caratteri. Esso include tutte le lettere (sia le maiuscole che le minuscole), le cifre e molti simboli speciali. Alcuni dei caratteri utilizzati nel Basic sono stati prelevati dal codice ASCII. Quando si parla di stringhe, si intende l'insieme dei caratteri ASCII e la loro rappresentazione numerica. Per esempio, il carattere A è rappresentato nel codice ASCII con il numero 65, il carattere B con il numero 66 ed il carattere 1 con il numero 49. La rappresentazione numerica della tastiera utilizza i numeri da 0 a 255.
Le numerose funzioni sulle stringhe fanno sì che il VIC sia un computer potente e versatile.
È abbastanza facile studiare dei programmi che "dialoghino" con l'operatore. I programmi Animal ed Eliza sono esempi di programmi che fanno largo uso di stringhe per instaurare un dialogo. Le stringhe servono anche a fornire notizie utili nel corso del programma, aiutando l'operatore nel suo compito.

Il modo migliore per imparare l'uso delle stringhe è fare programmi che le utilizzino. Eseguendo gli esercizi proposti in questo capitolo, cercate di capire bene ciò che state facendo prima di continuare. Alcuni esempi sono facili da interpretare e capire; altri sono, invece, più complicati. Tutti gli esempi devono essere provati sul VIC.

11.1 NOMI LECITI PER LE STRINGHE

Un nome di stringa è il nome di una variabile seguita dal simbolo $. Può essere lungo fino ad 85 caratteri incluso il $. Tuttavia, solo i primi due caratteri del nome sono presi in considerazione. Il primo carattere deve essere alfabetico.

Poiché solo i primi due caratteri sono utilizzati nel nome della variabile, bisogna fare attenzione alle solite convenzioni. Una stringa è delimitata da virgolette.

ESERCIZIO Controllate i nomi leciti per le variabili stringa.

```
NEW
ABCDE$="J"
AB123$="K"
PRINT ABCDE$,AB123$
K          K
PRINT AB$
K
A123456789$="S"
PRINT A1$
S
```

11.2 VARIABILI STRINGA AD INDICE

Non occorre dimensionare una variabile ad indice, se l'indice non è superiore a 10. Una variabile senza indice è distinguibile da una variabile con indice avente lo stesso nome. Per esempio, A$ è diversa da A$(0) oppure da A$(1).

ESERCIZIO Dimostrate l'indipendenza dei nomi delle variabili.

```
NEW
10 A$="A"
20 A$(0)="B"
30 A$(1)="C"
40 A$(2)="D"
50 PRINT A$;A$(0);A$(1);A(2)
RUN
ABCD
```

Notate che lo 0 è un indice lecito. Notate, inoltre, in linea 10, che A$ non è indicizzata ed è unica e distinguibile da una variabile con indice avente lo stesso nome. Non è stato necessario dimensionare (DIM) A$(0) e A$(2) poiché l'indice non supera 10.

ESERCIZIO Dimostrate che i vettori di stringhe con indice superiore a 10 devono essere dimensionati.

```
10 FOR I=1 TO 11
20 A$(I)="*"
30 NEXT I
40 END
RUN
?BAD SUBSCRIPT
 ERROR IN 20
```

Nell'esempio precedente, si deve adoperare per A$ l'istruzione DIM A$(11) poiché l'indice supera 10.

11.3 LUNGHEZZA MASSIMA DI UNA STRINGA

ESERCIZIO Determinate la lunghezza massima per una stringa.

```
NEW
FOR I=1 TO 255: A$=A$+"D":NEXT I 255
```

Si noti che una stringa può contenere al massimo 255 caratteri.

```
NEW
FOR I=1 TO 256:A$=A$+"D":NEXT I
?STRING TOO LONG
 ERROR
```

CAPITOLO DODICI

Operazioni sulle stringhe

12.1 ISTRUZIONI

Si può aumentare la lunghezza di una variabile stringa mediante la concatenazione. Concatenare significa raccordare.

ESERCIZIO Concatenate tre stringhe.

```
NEW
10 A$="V"
20 B$="I"
30 C$=A$+B$+"C"
40 PRINT C$
RUN
VIC
```

Notate come viene interpretato il simbolo + nel caso di valori numerici e nel caso di valori alfabetici.

ESERCIZIO Come funziona il simbolo + con i numeri e con le stringhe.

```
NEW
10 A$="123"
20 B$="456"
```

```
30 A=123
40 B=456
50 C$=A$+B$
60 C=A+B
70 PRINT C$,C
RUN
123456         579
```

Nella linea 50, la stringa B$ viene concatenata alla estremità destra della stringa A$, mentre, nella linea 60, viene effettuata una normale addizione.
Le stringhe devono essere manipolate in maniera differente rispetto ai valori numerici. Il segno + è l'unico operatore aritmetico che viene usato anche per le stringhe di caratteri. Tuttavia, gli operatori relazionali >, < ed il segno = lavorano nel seguente modo. Un carattere può essere minore, uguale o maggiore, rispetto al carattere a cui viene comparato, per quel che riguarda la sua rappresentazione numerica. Poiché la rappresentazione numerica di B è 66, mentre quella di A è 65, A risulta minore di B. Gli operatori logici booleani AND, OR e NOT non solo validi quando vengono utilizzati con stringhe.

ESERCIZIO Dimostrare che gli operatori logici non sono permessi con le stringhe.

```
C$=A$ OR B$
?TYPE MISMATCH
 ERROR
```

12.2 LE FUNZIONI

1. ASC La funzione ASC restituisce il valore numerico corrispondente al primo carattere ASCII incontrato in una stringa. Questa funzione può essere utile per trovare i valori ASCII di tutti i caratteri del VIC.

   ESERCIZIO Trovate il valore numerico di un carattere qualsiasi del VIC.

```
NEW
10 FOR I=1 TO 1000
20 INPUT A$
25 IF A$="" THEN STOP
30 A=ASC(A$)
40 PRINT A$,A
```

```
50 NEXT I
RUN
?
A          65
QUALSIASI CARATTERE

'RETURN'

BREAK IN 25
```

2. CHR$ La funzione CHR$ restituisce il carattere corrispondente all'intero dato. Il valore dell'intero deve essere compreso tra 0 e 255. La funzione CHR$ è l'inverso della funzione ASC, cioè, essa effettua la conversione inversa.

ESERCIZIO Visualizzate l'equivalente numerico di ogni carattere.

```
NEW
10 FOR I=0 TO 255
20 B$=CHR$(I)
30 PRINT I,B$
40 GET A$:IF A$="" THEN 40
50 NEXT I
RUN
0
```
qualsiasi tasto

Questo programma visualizzerà gli interi tra 0 e 255 ed il loro corrispondente carattere ASCII (si noti che qualche intero non avrà alcun carattere corrispondente).
Per fermare il programma battete STOP e RESTORE. Invece, per riattivare i caratteri maiuscoli battete:

```
?CHR$(142)
```

ESERCIZIO Dimostrate che CHR$ è l'inverso di ASC. Utilizzate, allo scopo, il precedente programma.

```
25 A=ASC(B$)
30 PRINT A;B$;I
RUN
0         0
```
qualsiasi tasto

.
.
.

La linea 20 converte il valore di I all'equivalente carattere del VIC. La linea 30 riconverte il carattere nel suo valore numerico. La linea 50 permette di controllare la velocità di visualizzazione dei valori.

ESERCIZIO Dimostrate che CHR$ è una stringa.

```
?CHR$(142)
NEW
10 B=CHR$(5)
20 PRINT B
RUN
?TYPE MISMATCH
 ERROR IN 10
```

Ricordate che CHR$ restituisce un carattere e che B è una variabile reale. Per ricordare il tipo di variabile da utilizzare, notate che ASC non ha il segno $, per cui il nome della variabile non deve avere tale segno. Invece CHR$ ha il segno $, perciò il nome della variabile deve averlo.

```
B$ = CHR$(40)
C = ASC("B")
```

3. LEFT$ La funzione LEFT$ è usata per selezionare i caratteri a sinistra in una stringa. Per utilizzare LEFT$, occorre dare il nome della variabile stringa ed il numero di caratteri scelti.

ESERCIZIO Selezionate caratteri da una stringa.

```
NEW
10 A$="ABCDE"
20 B$=LEFT$(A$,1)
40 PRINT B$
RUN
A
READY.
20 FOR I=0 TO 6
30 B$=LEFT$(A$,I)
50 NEXT I
RUN

A
AB
ABC
ABCD
```

```
ABCDE
ABCDE
OK
```

Attenzione! Non vi è messaggio di errore se I è maggiore di 5, cioè maggiore del numero di caratteri di A$. In questo caso viene restituita l'intera stringa. Si noti che quando I era zero, non è stato fornito alcun carattere a B$.

4. RIGHT$ La funzione RIGHT$ è usata per selezionare i caratteri a destra in una stringa. Essa funziona esattamente come la funzione LEFT$, eccetto che considera i caratteri alla destra. Per dimostrare ciò, basta cambiare LEFT$ con RIGHT$ nell'esempio precedente.
Attenzione: Non vi è messaggio di errore se si oltrepassa la lunghezza della stringa. Vedere istruzione LEFT$.

5 MID$ La funzione MID$ è usata per selezionare qualsiasi carattere o gruppo di caratteri da una stringa. MID$ è la funzione più potente tra le tre. Per utilizzare MID$ basta fornire il nome della variabile, la posizione del primo carattere ed il numero di caratteri da estrarre. MID$ con solo due argomenti restituisce il resto della stringa.

ESERCIZIO Selezionate caratteri alla sinistra di una stringa. Utilizzate il programma dell'esercizio precedente.

```
30 B$=MID$(A$,1,I)
RUN

A
AB
ABC
ABCD
ABCDE
ABCDE
READY.
```

Ora selezionate caratteri alla destra della stringa.

```
5 J=0
20 FOR I=6 TO 0 STEP -1
25 J=J+1
30 B$=MID$(A$,I,J)
RUN
E
DE
```

```
CDE
BCDE
ABCDE
?ILLEGAL QUANTITY
 ERROR IN 30
READY.
```

Attenzione! Non vi è messaggio di errore se I o J o ambedue sono maggiori del numero di caratteri in A$.

6. LEN La funzione LEN conta i caratteri in una stringa, restituendone il numero.

ESERCIZIO Usate la funzione LEN per scindere una stringa in due stringhe di lunghezza uguale (o quasi).

```
NEW
10 INPUT A$
20 A=LEN(A$)
30 B=INT(A/2)
35 IF A=1 GOTO 50
40 B$=LEFT$(A$,A-B)
50 C$=RIGHT$(A$,B)
60 PRINT B$,C$
RUN
?
```

Battete una stringa di vostra scelta.

7. STR$ La funzione STR$ viene usata per convertire un numero in una stringa. Se il numero ha più di 9 cifre, viene effettuato l'arrotondamento ed il valore è convertito in forma esponenziale.

ESERCIZIO Convertite un numero in una stringa.

```
A=1234567885123
PRINT STR$(A)
1.23456789E+12
```

STR$ può essere utilizzata per allineare i punti decimali in un programma di gestione del conto corrente.

ESERCIZIO Scrivete un programma per l'allineamento dei punti decimali di una serie di numeri.

```
NEW
10 DATA 10.03,100.25,.25,6.13,0
20 READ A$
30 IF A$="0" GOTO 120
40 FOR I=1 TO LEN(A$)
50 IF MID$(A$,I,1)="." GOTO 70
60 NEXT I
70 FOR J=1 TO 10-I
80 A$=" "+A$
90 NEXT J
100 PRINT A$
110 GOTO 20
120 END
RUN
   10.03
  100.25
     .25
    6.13
```

8. VAL La funzione VAL è usata per convertire una stringa, avente caratteri numerici, in un valore numerico. Questa funzione è l'inverso della funzione STR$. Se vi sono dei caratteri alfabetici nella stringa, nella conversione verranno considerati solo le cifre alla sinistra del primo carattere alfabetico.

ESERCIZIO Convertite una stringa nel suo equivalente numerico.

```
NEW
10 A$="1234"
20 B$="A1234"
30 C$="123B4"
40 A=VAL(A$)
50 B=VAL(B$)
60 C=VAL(C$)
70 PRINT A;B;C
RUN
1234  0  123
```

La variabile A$ è stata convertita come previsto. La variabile B è uguale a 0, poiché il primo carattere di B$ è di tipo alfabetico. La stringa C$ è convertita fino al primo carattere alfabetico.

CAPITOLO TREDICI

Rappresentazione ed elaborazione dei dati

Se non avete ancora padronanza degli operatori di relazione, dei sistemi numerici, del codice dei caratteri, degli operatori logici e di come i dati sono memorizzati, vedere appendice B.

13.1 PROGRAMMI PER LA CONVERSIONE DEL SISTEMA NUMERICO

Il primo argomento da considerare riguarda la conversione da un sistema numerico ad un altro. I seguenti esercizi sviluppano i programmi necessari a tale scopo.

ESERCIZIO Scrivete un programma per visualizzare la rappresentazione binaria di un numero decimale. Un listato completo del programma Dec-Bin si trova in appendice A.

```
NEW
90 REM SCRIVE IL NUMERO BINARIO
100 INPUT "SCRIVI IL N. DECIMALE";R
430 R$=""
440 GOSUB 2000
500 PRINT "DECIMALE";R;"="
510 PRINT "BINARIO";R$
```

```
520 PRINT
530 GOTO 100
```

Il programma principale richiede in ingresso un numero decimale. La subroutine, che ha inizio alla linea 2000, esegue il lavoro vero e proprio. Poi il programma principale visualizzerà il risultato. Infatti il programma principale passa alla subroutine il valore numerico da convertire (contenuto in R) ed otterrà in R$ il risultato.

```
2000 REM CONVERSIONE DEL NUMERO IN STRINGA BINARIA
2010 C$(0)="0":C$(1)="1"
2020 R2=R
2030 T2=INT(R2/2)
2040 S=R2-T2*2
```

S è il resto e T2 è il quoziente.

```
2050 R$=C$(S)+R$
2060 R2=T2
2065 IF LEN(R$)>16 THEN R$=RIGHT$(R$,16): RETURN
2070 IF T2<>0 THEN 2030
```

Il vecchio quoziente T2 diventa il nuovo dividendo R2. Tale operazione è ripetuta finché tutte le cifre binarie sono state assemblate in R$.

```
2080 RETURN
```

Salvate il programma con il nome DEC-BIN. Verrà utilizzato più tardi per descrivere l'esecuzione di operazioni logiche.

```
RUN
SCRIVI IL N. DECIMALE?
38
DECIMALE 38 =
BINARIO 100110
SCRIVI IL N. DECIMALE?
```

Si provi con qualche numero decimale. Questo programma può essere usato per controllare la propria capacità di convertire a mano numeri dal sistema decimale a quello binario.

```
SCRIVI IL N. DECIMALE?
```

Per terminare questo programma battete 1E40.

ESERCIZIO Convertite un numero binario, contenuto in una stringa, in numero decimale. Caricate il programma DEC-BIN se non è già in memoria. Si utilizzerà una parte di esso. Un listato completo di BIN-DEC è in appendice A.

```
90 REM SCRIVE IL NUMERO DECIMALE
100 PRINT "SCRIVI IL N. BINARIO"
105 INPUT N$
106 IF N$="" THEN STOP
110 T1$=N$
120 GOSUB 1100
130 IF T<0 THEN 100
140 PRINT N$;" BINARIO"
145 PRINT "=";T;"DECIMALE"
150 GOTO 100
```

Il programma principale considera la stringa con il numero binario e la passa alla subroutine di conversione (1100). Se la conversione è eseguibile, viene visualizzato il valore.

```
1100 REM CONVERSIONE BINARIO-DECIMALE
1105 T=0
1110 L1=LEN(T1$)
1120 FOR I1=1 TO L1
1125 T=T*2
1130 IF MID$(T1$,I1,1)="1" THEN T=T+1: GOTO 1150
1140 IF MID$(T1$,I1,1)<>"0" THEN T=-1:RETURN
1150 NEXT I1
1160 RETURN
```

La subroutine converte la serie di caratteri binari in numero decimale. Viene restituito un *flag* di errore per qualsiasi carattere che non sia lo zero o l'uno.

```
RUN
SCRIVI IL N. BINARIO
?
101
101 BINARIO
= 5 DECIMALE
```

Si provi con qualche numero binario. Per terminare questo programma battete " ". Salvate il programma con il nome BIN-DEC.

13.2 PROGRAMMI DI OPERAZIONI LOGICHE

Le tre funzioni logiche disponibili con il Basic Commodore sono:

a) AND
b) OR
c) NOT

Se non siete ancora a conoscenza di questa terminologia, si veda in appendice B. Ciascuna di queste funzioni, ad eccezione del NOT, ha due valori di ingresso ed un solo valore di uscita (risultato). La funzione NOT ha un valore di ingresso ed uno di uscita. A questi valori ci si riferisce con le espressioni vero e falso. Il VIC utilizza un numero diverso da zero per il *vero* e lo zero per il *falso*.

ESERCIZIO Visualizzate il valore della funzione logica eseguita con vari dati. Caricate il programma BIN-DEC ed aggiungete le seguenti linee (un listato completo si trova in appendice A):

```
90 REM FUNZIONI LOGICHE
140 NE=T
145
150 PRINT"SCRIVI LA FUN. LOGICA"
160 INPUT"(A,O,N)";OP$
190 F$=LEFT$(OP$,1)
200 IF F$<>"A" AND F$<>"O" AND F$<>"N" THEN 150
```

Considera l'operatore logico scelto (AND,OR,NOT), controllando la validità del primo carattere.

```
210 TW$=""
220 IF F$="N" THEN 420
```

Poiché NOT necessita di un solo argomento, non viene considerato l'ingresso del secondo argomento e la sua conversione.

```
300 PRINT"SCRIVI IL SECONDO NUM BINARIO"
305 INPUT TW$
310 T1$=TW$
320 GOSUB 1100
330 IF T<0 THEN 300
```

Assicurarsi che il numero inserito sia un valore binario lecito.

```
340 TWO=T
400 IF F$="A" THEN R=NE AND TWO
410 IF F$="O" THEN R=NE OR TWO
420 IF F$="N" THEN R=NOT NE
```

Effettua l'operazione logica desiderata sui parametri in ingresso.

```
500 IF F$="N" THEN PRINT OP$;" ";N$;" ="
505 IF F$<>"N" THEN PRINT N$;" ";OP$;" ";TW$;" ="
510 PRINT R$
```

Visualizza l'ingresso e l'operazione con il relativo risultato.
Salvate il programma con il nome Logic. Si provi con vari esempi.

```
RUN
SCRIVI IL N. BINARIO
?
101
SCRIVI LA FUN. LOGICA
(A,O,N)?
A
SCRIVI IL SECONDO NUM
BINARIO
11
101 A 11 =
1
SCRIVI IL N. BINARIO
?
```

Si controlli il programma con altri valori ed altre operazioni. Il programma termina rispondendo con " " alla richiesta del primo numero binario.

Variabili con indice

Così come una variabile normale, anche la variabile con indice dà un nome ad una locazione di memoria nel computer. Queste locazioni possono essere rappresentate da una serie di piccole caselle postali contigue, dove la variabile con indice è il nome della casella ed il valore della stessa è il contenuto della casella. Un'altra rappresentazione di queste locazioni di memoria potrebbe essere una serie di righe e colonne. Nell'esempiò seguente, le locazioni di memoria sono le caselle, il contenuto della memoria sono i numeri entro le caselle e l'indirizzo o nome della locazione di memoria è la variabile con indice posta sulla sinistra.

A(0)	1
A(1)	3
A(2)	10
A(3)	211

Il valore di A(0) è 1
Il valore di A(3) è 211
Il nome della locazione di 3 è A(1)

Un insieme di elementi organizzati in una colonna o in una riga è detto *vettore monodimensionale.* I vettori bidimensionali hanno sia le righe che le colonne e vengono detti *matrici.* Gli indici definiscono la riga e la colonna.

	colonna 1	colonna 2	colonna 3
riga 1	A(1,1)	A(1,2)	A(1,3)
riga 2	A(2,1)	A(2,2)	A(2,3)

Le variabili indice consentono di ordinare facilmente un insieme di N elementi di una tabella. Con il Basic del VIC, le variabili numeriche e le variabili stringa possono essere indicizzate. Usando le variabili con indice, è possibile assegnare tanti valori alla stessa variabile, cambiando semplicemente l'indice. Il motivo di questa semplificazione è che le variabili con indice sono l'ideale nei cicli FOR...NEXT. Verranno fatti diversi esempi in cui il valore dell'elemento della variabile con indice è l'indice stesso.

ESERCIZIO Introduzione e visualizzazione di variabili normali e di vettori.

```
NEW
10 DATA 5,7,8,11,12
20 READ A1,A2,A3,A4,A5
30 PRINT A1;A2;A3;A4;A5
RUN
5  7  8  11  12
```

Senza un vettore, occorrerebbe scrivere separatamente ogni variabile. Non vi è difficoltà per cinque variabili, ma se queste fossero 100? Il seguente listato utilizza un vettore per fare la stessa cosa.

```
20 FOR I=1 TO 5
30 READ A(I)
40 PRINT A(I);
50 NEXT I
RUN
5  7  8  11  12
```

14.1 INDICI LECITI

I valori leciti per gli indici sono numeri interi non negativi. Possono essere utilizzate allo scopo anche le variabili, le formule, le funzioni e le costanti.

ESERCIZIO Dimostrate che le variabili, le funzioni e le formule sono degli indici leciti.

```
NEW
10 A=2
20 B=3
```

```
30 C=5
40 FOR I=1 TO 10
50 D(I)=I
60 NEXT I
70 PRINT D(A*B-C)
80 PRINT D(INT(C/A))
90 PRINT D(C-INT(B/A)+A)
RUN
1
2
6
OK
```

14.2 LE DIMENSIONI

Una variabile con indice non necessita della corrispondente istruzione DIM finché il valore dell'indice non supera il 10 (ciò vale anche nel caso di variabili a più indici). Il valore massimo per l'indice rimane 10, anche nel caso si utilizzi l'indice 0. Perciò, utilizzando anche lo zero, si hanno a disposizione 11 elementi per ogni variabile con indice, senza bisogno di utilizzare l'istruzione DIM per quella variabile.

ESERCIZIO Trovate il limite per le variabili numeriche con indice non dimensionate.

```
NEW
10 FOR J=1 TO 20
20 FOR I=1 TO 20
30 A(I,J)=I
40 PRINT A(I,J);I;J
50 NEXT I
60 NEXT J
70 END
RUN
1  1  1
2  2  1
         .
         .
         .
10  10  1
?BAD SUBSCRIPT
 ERROR IN 30
```

Senza l'istruzione DIM per la variabile A, il programma si fermerà quando I o J supereranno il valore 10.

ESERCIZIO Visualizzate una variabile con indice non dimensionata iniziando con il valore 0.

```
NEW
10 FOR I=0 TO 10
20 A(I)=I
30 PRINT A(I);I
40 NEXT I
RUN
0  0
1  1
2  2
        .
        .
        .
10  10
READY.
```

La seguente lista di istruzioni DIM è il massimo numero di variabili con indice con un VIC da 5 K, senza incorrere nel messaggio di errore ?OUT OF MEMORY ERROR. Questi valori sono stati trovati utilizzando il computer nel modo diretto, cioè senza alcun programma in memoria. Ricordarsi di battere NEW prima di ogni prova.

```
DIM
DIM A(713)        DIM A(25,26)      DIM A(8,8,7)
DIM A%(1785)      DIM A%(41,44)     DIM A%(11,11,11)
DIM A$(1190)      DIM A$(34,33)     DIM A$(10,9,9)
```

Le variabili con indice possono essere combinate con le istruzioni READ...DATA per inserire i dati di una tabella.

ESERCIZIO Utilizzazione di READ e di DATA per riempire un vettore stringa. (Un vettore stringa è un insieme ordinato di stringhe.)

```
NEW
99 DIM MN$(12)
100 FOR I=1 TO 12
110 READ MN$(I)
120 DATA"GEN","FEB","MAR","APR","MAG","GIU"
125 DATA"LUG","AGO","SET","OTT","NOV","DIC"
```

```
130 PRINT MN$(I)
140 NEXT I
RUN
GEN
FEB
.
.
.
DIC
```

Questo programma inizializza il vettore MN$ con i nomi dei mesi e li stampa. Per verificare che il quinto mese è maggio, provate:

```
PRINT MN$(5)
MAG
```

ESERCIZIO Utilizzate READ e DATA per riempire un vettore numerico. (Un vettore numerico è un insieme ordinato di valori.)

```
99 DIM ND(12)
110 READ ND(I)
120 DATA 31,28,31,30,31
125 DATA 30,31,31,30,31,30,31
130 PRINT ND(I)
RUN
31
28
.
.
.
31
```

Questo programma inserisce il numero dei giorni di ogni mese in un vettore ND e li visualizza. Per verificare che il quinto mese ha 31 giorni, provate con:

```
PRINT ND(5)
31
```

CAPITOLO QUINDICI

Schema di un programma

Il ciclo di sviluppo di un programma è stato descritto nel capitolo 1. Un passaggio importante in questo ciclo è quello dello sviluppo della soluzione. Nei capitoli dal 2 al 14 sono state commentate varie caratteristiche del Basic Commodore. Non è stata richiesta la soluzione di veri e propri problemi, poiché gli esercizi erano solo degli esempi. Nei due capitoli successivi, riguardanti il colore e il suono del VIC, verranno sviluppati dei programmi, utilizzando gli algoritmi imparati precedentemente. Invece di fornire dei programmi già pronti, verrà solo descritto il metodo risolutivo del problema proposto.
Lo schema di un programma si divide in due parti:

1. la procedura (passi di programma);
2. i dati da elaborare.

Queste due parti devono essere considerate insieme. È possibile che non sia necessario adoperare nei programmi il colore od il suono, tuttavia è bene esercitarsi ugualmente con gli esempi proposti. Essi, infatti, usano tecniche utili in ogni programma. Il procedimento per lo sviluppo di un programma può essere suddiviso in sette passaggi.

1 Definire il problema.
2 Stendere, a grandi linee, la soluzione passo per passo.
3 Specificare i dati che occorrono alla procedura descritta nel passo **2.**
4 Decidere ciò che dovrà essere eseguito dal computer e quello che dovrà fare l'utente.

5 Ritoccare la soluzione trovata, migliorandola se possibile.
6 Ridefinire i dati, già specificati in **3**, per adattarli alla soluzione indicata in **5**.
7 Se la soluzione ed i dati ottenuti sono soddisfacenti, passare alla fase del listato; altrimenti, ritornare al punto **4**.

15.1 PSEUDOLISTATO

Lo pseudolistato è più leggibile e comprensibile del Basic o di un diagramma di flusso. Esso è usato per dare delle indicazioni sulla struttura del programma. La parte in italiano dello pseudolistato descrive ciò che deve essere fatto e non come farlo. Invece di utilizzare dei simboli e dei tratti di unione (come nel diagramma di flusso), lo pseudolistato utilizza delle parole chiave e diverse marginature dei periodi per visualizzare l'organizzazione di un processo. Lo pseudolistato è più leggibile perché non vi sono salti (GOTO). Perciò, esso può essere letto dall'inizio alla fine senza interruzioni.
Lo pseudolistato non è un linguaggio di programmazione poiché non funziona su alcun computer. Tuttavia, esso verrà utilizzato per descrivere il passaggio da una serie di istruzioni generiche alla soluzione in Basic del programma. Questo passaggio graduale aiuta ad identificare ciò che è realmente richiesto, semplificando la stesura di un programma. È facile trascrivere lo pseudolistato in Basic od in altri linguaggi di programmazione. Il compito più difficile è decidere ciò che il programma deve fare. Lo pseudolistato aiuta a scegliere ciò di cui si ha bisogno.
Ripassate velocemente le definizioni delle diverse istruzioni lecite dello pseudolistato. Proseguite poi al capitolo 16, dove queste istruzioni sono usate per sviluppare un programma per la selezione dei colori. L'esempio dovrebbe aiutare a capire le definizioni.

1. *Forma generale* La forma generale per uno pseudolistato è composta da parole chiave e da un testo. Le parole chiave sono tutte identificate da lettere maiuscole. Il testo è scritto sia con lettere maiuscole che minuscole. Le parole chiave utilizzate sono: SELECT, ENDSELECT, OTHERWISE, REPEAT ed ENDREPEAT.

2. *Sequenza* Una sequenza di azioni è descritta usando solo il testo, senza utilizzare le parole chiave. Per esempio, l'istruzione Visualizza i dati errati è un testo lecito in uno pseudolistato per indicare una azione o sequenza di azioni necessaria alla risoluzione di parti di un problema.

3. *Condizionale* Un'istruzione condizionale dello pseudolistato permette di scegliere tra diverse azioni. A tale scopo, vengono usate le parole chiave SELECT, OTHERWISE ed ENDSELECT. La forma generale è:

```
SELECT sotto la condizione... (specificare)
(condizione tipo 1)
    pseudoistruzioni
(condizione tipo 2)
    pseudoistruzioni
....................
(condizione tipo N)
    pseudoistruzioni
(OTHERWISE)
    pseudoistruzioni
ENDSELECT
```

La forma generale è così interpretata: se si verifica la condizione 1, eseguire le pseudoistruzioni specificate nel caso 1, cioè, tutte le istruzioni tra la condizione tipo 1 e la condizione tipo 2. Similmente, se si verifica la condizione I, eseguire le pseudoistruzioni specificate nel caso I, cioè, tutte le pseudoistruzioni tra la condizione tipo I e la condizione tipo I + 1. Se non si verifica nessuna delle condizioni, eseguire tutte le pseudoistruzioni specificate da OTHERWISE, cioè tutte le pseudoistruzioni tra OTHERWISE ed ENDSELECT. A prescindere dalla scelta, il controllo verrà trasferito alle pseudoistruzioni successive a ENDSELECT.

Vi sono due casi elementari:

```
SELECT sotto la condizione... (specificare)
(condizione vera)
    pseudoistruzioni
ENDSELECT
```

Questo caso elementare significa che se la condizione è vera viene fatto qualcosa. Se la condizione è falsa, non viene fatto nulla. L'altro caso elementare è:

```
SELECT sotto la condizione...
(condizione vera)
    pseudoistruzioni
(OTHERWISE)
    pseudoistruzioni
ENDSELECT
```

Questa forma significa che se la condizione è vera, viene fatta una determinata cosa, se, invece, è falsa, viene fatta un'altra cosa. In ambedue i casi, la successiva cosa da fare è indicata dopo ENDSELECT.

4. *Ripetizioni* La pseudoistruzione REPEAT permette di specificare la ripetizione delle stesse operazioni un numero determinato di volte. Le azioni possono essere eseguite molte volte, qualche volta, una volta o mai. Tutto dipende dalla condizione. Le parole chiave usate allo scopo sono REPEAT e ENDREPEAT. La forma generale è:

```
REPEAT fino alla condizione...
    pseudoistruzioni
ENDREPEAT
```

Tutte le pseudoistruzioni comprese tra REPEAT e ENDREPEAT vengono ripetute finché non si verifica la condizione indicata.

5. *Indentazione* Come si può notare, tutte le linee interne ad una determinata istruzione, come nel caso della selezione o della ripetizione, hanno la stessa indentazione. L'indentazione evidenzia la correlazione delle varie linee con le parole chiave aventi inizio nella colonna immediatamente precedente. Come esempio si può descrivere un semplice calcolo di preventivi. Vi sono diversi lavori da stimare. Per alcuni ci si può servire di un testo, per altri è necessario l'intervento del caporeparto.

```
REPEAT finché tutti i preventivi non siano visualizzati
    Preparare la copertina dei preventivi
    SELECT tra i tipi di preventivi
    (il lavoro è di tipo normale)
        Prendere il libro
        Preparare il preventivo con l'aiuto del libro
        Visualizzare il preventivo
    (il lavoro è di tipo speciale)
        Chiedere al caporeparto di preparare il preventivo
        Visualizzare il preventivo
    ENDSELECT
ENDREPEAT
```

Il primo livello di indentazione è utilizzato da REPEAT e da ENDREPEAT. Tutto ciò che è indentato oltre il primo livello è interno all'istruzione REPEAT. Le parole chiave SELECT ed ENDSELECT sono al secondo livello di indentazione e, similmente a prima, tutto ciò che è indentato oltre il secondo livello è interno all'istruzione SELECT. La linea il lavoro è di tipo normale è al secondo livello. Essa segna l'inizio di una

delle due scelte. Le linee seguenti a (il lavoro è di tipo normale) che sono indentate a destra del secondo livello, fanno parte di quella scelta. La successiva linea nel secondo livello, (il lavoro è di tipo speciale), indica la fine della prima scelta.
Questo è un esempio relativamente semplice, ma è evidente che sarebbe meno comprensibile se fosse visualizzato senza indentazione.

```
REPEAT finché tutti i preventivi non siano visualizzati
Preparare la copertina dei preventivi
SELECT tra i tipi di preventivi
(il lavoro è di tipo normale)
Prendere il libro
Preparare il preventivo con l'aiuto del libro
Visualizzare il preventivo
(il lavoro è di tipo speciale)
Chiedere al caporeparto di preparare il preventivo
Visualizzare il preventivo
ENDSELECT
ENDREPEAT
```

È necessario utilizzare un tipo di indentazione che aiuti la compresione della struttura di uno pseudolistato. Il numero degli spazi di margine è arbitrario. Tre spazi rappresentano un buon compromesso. È difficile notare le differenze nei livelli di istruzioni se il margine è inferiore a tre spazi. Più di tre spazi sono uno spreco inutile di carta e rendono complesso il contenere in una linea di 65-75 caratteri uno pseudolistato con molti "nidi" di istruzioni.

15.2 DESCRIZIONE DEI DATI

La descrizione dei dati è parallela alla descrizione delle procedure di uno pseudolistato. La descrizione dei dati comprende la sequenza, la selezione e la ripetizione. I dati sono descritti utilizzando un testo e qualche carattere speciale. Non vi è bisogno di parole chiave per descrivere i dati. Questa descrizione è a volte detta *dizionario dei dati.*

1. *Sequenza* Una sequenza di dati è descritta con un testo ed il segno +. Per esempio:

Colore = Colore sfondo + Colore bordo + Colore carattere

viene letto come: Colore è uguale a Colore sfondo AND Colore bordo

AND Colore carattere. Tutti e tre i dati sono necessari per ottenere il dato Colore.

2. *Selezione* Una selezione di dati è una scelta tra diversi possibili elementi. I simboli speciali [, I e] vengono impiegati per indicare una scelta. Il simbolo [segna l'inizio delle scelte. Il simbolo I separa le varie scelte ed il simbolo] segna la fine delle scelte possibili. La forma generale è la seguente:

Questi dati = [Primo tipo I
Secondo tipo I
Terzo tipo]

Essa viene letta come: Questi dati appartengono al Primo tipo OR al Secondo tipo OR al Terzo tipo.
Così come esistono due casi speciali di selezione dei passi di procedura, esistono due casi speciali di selezione di dati. Il primo caso è una semplice scelta tra due alternative.

Scelta = [Uno I Due]

Ciò si può leggere come: Scelta è Uno OR Due. La sbarra è usata per separare i due dati. Il secondo caso speciale è:

Opzione = [Solo una]

Questo elemento del dizionario dei dati significa: Opzione è solo una OR niente.

3. *Ripetizione* L'ingresso di un elemento del dizionario dei dati che specifica la ripetizione indica che un elemento deve essere introdotto un numero determinato di volte. I dati possono apparire più volte, qualche volta, una volta o mai. Il numero di volte dipende dai limiti visualizzati e dai dati attuali. La forma generale è:

Nome = 0 <Titolo> 5+Nome+Cognome

Ciò significa Nome è uguale ad un titolo tra 5 possibili AND Nome AND Cognome. Il primo numero rappresenta il limite inferiore, i simboli < e > indicano l'inizio e la fine della ripetizione e l'ultimo numero rappresenta il limite superiore. Si può pensare ai simboli summenzionati come all'oggetto della ripetizione.

ESEMPIO Descrivete una stringa di caratteri del VIC.

Stringa caratteri VIC = 0 carattere 255
Carattere = 8 cifra binaria 8
Cifra binaria = [0/1]

Questo insieme di ingressi per il dizionario dei dati specifica che una stringa di caratteri del VIC è composta da 0 a 255 caratteri, che ogni carattere è esattamente composto da 8 cifre binarie e che ogni cifra binaria può essere uno 0 od un 1.

15.3 IMPLEMENTAZIONE TIPICA DEL BASIC

Le istruzioni di uno pseudolistato devono essere tradotte in un linguaggio di programmazione. Nel VIC questo può essere o il Basic o il linguaggio macchina. Il linguaggio macchina non verrà trattato in questo libro. Lo pseudolistato può essere tradotto in Basic in diversi modi.

1. *Sequenza* Una sequenza di istruzioni può essere un'istruzione semplice come l'assegnazione, l'ingresso o l'uscita.

ESEMPIO Istruzioni in sequenza

```
A=5
READ B
PRINT#1,C$
```

2. *Selezione* Le istruzioni di controllo come ad esempio IF, ON X GOTO... oppure ON X GOSUB..., vengono usate per implementare la selezione.

ESEMPIO Scelta tra quattro alternative.

```
ON I GOTO 100,200,300,400

100 IF A=0 THEN B=5:GOTO 200
110 B=6
200 ...

300 IF B=5 THEN C=A
```

3. *Ripetizione* La ripetizione può essere fatta in diversi modi. Se la ri-

petizione è regolare, è possibile adoperare un ciclo FOR... NEXT.

ESEMPIO Ciclo regolare per la ripetizione.

```
100 FOR I=1 TO 50
110 A(I)=I
120 NEXT I
```

Se la ripetizione è condizionale, come nella ricerca di un dato in una tabella, può essere eseguita con una istruzione IF ed un GOTO.

ESEMPIO Ripetizione condizionale.

```
100 I=0
110 I=I+1
120 IF I>LIMIT THEN 200
130 IF A(I)=TEST THEN 300
140 GOTO 110
200 REM NON C'E'
300 REM TROVATO L'ELEMENTO I
```

Se la ripetizione deve essere effettuata finché l'operatore non ne segnala la fine, si può operare in due modi.

ESEMPIO L'operatore batte i tasti STOP e RESTORE per terminare.

```
100 INPUT D
 ...
 ...
 ...
200 GOTO 100
```

ESEMPIO L'operatore batte la parola FINE per terminare.

```
100 INPUT D$
110 IF D$="FINE" THEN STOP
120 D=VAL(D$)
 ...
 ...
 ...
200 GOTO 100
```

Nei successivi capitoli sul colore ed il suono verranno forniti più esempi per mostrare come tradurre uno pseudolistato in Basic.

CAPITOLO SEDICI

Colore

In questo capitolo si utilizzeranno le tecniche di sviluppo di programmi descritte nel capitolo 15. Questo programma presenterà le combinazioni di colori disponibili con il VIC.

16.1 ISTRUZIONI DEL PROBLEMA

Il problema da risolvere è progettare un programma che visualizzi sul VIC i colori scelti dall'utente. Al primo livello (il più generale) lo pseudolistato potrebbe essere così descritto:

```
Prepara i dati necessari
REPEAT finché l'operatore non trova la combinazione desiderata
    Scegli il colore
    Disponi il VIC con quei colori
ENDREPEAT
```

Questa procedura necessita di un ingresso per specificare il colore ed inoltre dell'informazione su come predisporre il VIC per un colore particolare.

Bisogna rispondere a diversi quesiti prima di proseguire con il passo successivo (rifinitùra della soluzione). Questi quesiti aiuteranno a chiarire il

progetto, determineranno ciò che dovrà essere eseguito dal programma e quello che dovrà fare l'utente.

1. Quali sono le scelte sui colori che è possibile operare? Si assume che sia possibile selezionare il colore del bordo, dello sfondo e del carattere.
2. Come verranno gestiti gli errori? È lecita la richiesta di un carattere nero su sfondo nero? Si assume che la scelta di un colore non esistente o di uno "invisibile" (rosso su rosso) dia un messaggio di errore. Dopo il messaggio, verrà visualizzata una lista di colori leciti per questo tipo. Tipo è sfondo, bordo o carattere.
3. Come devono essere specificati i colori? Il VIC possiede dei tasti contrassegnati con il nome dei colori. Questo è un buon metodo per la selezione dei colori. Alcuni potrebbero preferire battere direttamente il nome del colore scelto oppure una sua abbreviazione. Verranno forniti tutti e tre i metodi di scelta del colore.

16.2 RIFINITURA DELLA SOLUZIONE

Basandosi sul primo livello di progetto e sulle risposte ai quesiti 1, 2, 3, si può iniziare a sviluppare e rifinire il progetto. Imbattendosi in quesiti sempre più dettagliati, aumenterà ulteriormente il grado di rifinitura del progetto. Per iniziare la descrizione del secondo livello occorre rifinire le istruzioni del primo livello (Scegli il colore). È possibile scindere tale istruzione nelle seguenti:

```
Scegli il colore dello sfondo
Scegli il colore del bordo
Scegli il colore del carattere
```

Queste tre linee sono equivalenti alla linea originale e rappresentano un piccolo passo verso la descrizione completa del problema da risolvere.
I dati richiesti possono essere ora chiariti. Il programma deve essere in grado di scegliere tra gli ingressi il colore per lo sfondo, per il bordo e per il carattere. Inoltre, deve essere in grado di predisporre il VIC con tali colori.
Finora sono state utilizzate solo le istruzioni dello pseudolistato spiegato nel capitolo 15. Non vi era bisogno di parole chiave. Avviandosi verso il terzo livello, occorrerà fare uso della ripetizione e della selezione per rifinire ulteriormente questa parte di soluzione. Nel terzo livello verranno usate le informazioni già note, più la risposta 2, per rifinire i tre

Scegli... del livello due. Perciò, l'istruzione Scegli il colore dello sfondo può essere così descritta:

```
REPEAT finché non sia inserito il colore lecito per lo sfondo
    Inserisci il colore dello sfondo
    SELECT il colore lecito per lo sfondo
    (colore illecito per lo sfondo)
        Visualizza il messaggio di errore
        Visualizza i colori leciti per lo sfondo
    (colore lecito per lo sfondo)
        Salva il colore lecito per lo sfondo
    ENDSELECT
ENDREPEAT
```

Questa sequenza di pseudolistato descrive, con più dettagli, ciò che è necessario per Scegli il colore dello sfondo e mostra, inoltre, quello che si dovrà fare in caso di errore.
I dati necessari allo scopo sono una lista di colori leciti per lo sfondo. Nel quarto livello, utilizzando la risposta alla terza domanda, è possibile rifinire (essere più precisi) l'istruzione Inserisci il colore dello sfondo.

```
Richiedi all'operatore il colore dello sfondo
Prendi le informazioni sul colore della tastiera
```

Poiché il VIC gestisce numeri e avrà bisogno di un numero per assegnare il colore, occorre convertire i nomi e le abbreviazioni dei colori nel loro numero corrispondente. Il controllo dei dati leciti può essere effettuato, in parte mediante ricerca su tabella e, d'altra parte, utilizzando un numero. Fissati questi criteri, si può ampliare l'istruzione Prendi le informazioni sul colore dalla tastiera nel seguente pseudolistato:

```
Leggi la stringa di caratteri dalla tastiera
SELECT il contenuto della stringa
(La stringa è un numero)
    Assegna al colore un numero
    SELECT il numero
    (Il numero non è lecito)
        Assegna al numero del colore il valore illecito standard
    ENDSELECT
(La stringa è l'abbreviazione di un colore)
    Assegna il numero corrispondente all'abbreviazione del colore
(La stringa è il nome per intero del colore)
    Assegna il numero corrispondente al nome per intero del colore
(OTHERWISE)
```

```
    Assegna al numero del colore la condizione illecita
ENDSELECT
```

Si noti che OTHERWISE "cattura" solo alcuni fra i colori illeciti. I rimanenti saranno poi intercettati nella sezione di controllo per il carattere "invisibile".
Le tre informazioni necessarie per questi passaggi sono: la richiesta all'operatore, una tabella di abbreviazioni ed una tabella di nomi per intero dei colori.
Il SELECT condizionale, basato sul colore illecito dello sfondo, può essere eseguito in diversi modi. Un metodo di controllo relativamente semplice è quello di confrontare il nome del colore con la lista dei nomi di colore leciti. L'istruzione Assegna il numero corrispondente al nome per intero del colore può essere rifinita sostituendola con un ciclo REPEAT che effettui una ricerca nella tabella delle abbreviazioni.

4. È preferibile restringere la scelta dei colori agli otto colori fondamentali, oppure poter considerare tutte le combinazioni di colori disponibili per lo sfondo?

Si può generalizzare la procedura consentendole di accettare tutti i colori possibili. Naturalmente, i colori dello sfondo aggiuntivi non avranno un tasto a loro dedicato, ma potranno essere inseriti mediante un numero, con un nome intero (ad esempio AZZURRO) oppure con una abbreviazione (ad esempio AZZ). Questi colori aggiuntivi saranno illeciti se utilizzati come colori del bordo o del carattere. Perciò i dati devono contenere le informazioni se i colori per ogni carattere tipografico siano leciti o no. La seguente procedura di ricerca nell'ambito di una tabella può essere utilizzata sia per la tabella dei nomi interi, sia per la tabella delle abbreviazioni.

```
Supponi che il colore non sia stato trovato
Posizionati sul primo colore della tabella
REPEAT finché non siano stati controllati tutti i colori OR
        finché il colore non sia stato trovato
    SELECT il colore inserito
    (Sì, il colore è nella tabella)
        Il colore è stato trovato
    ENDSELECT
    Avanza sul colore successivo
ENDREPEAT
```

Gli pseudolistati per ottenere il colore dello sfondo e quello del carattere sono simili ad eccezione del nome del colore. Poiché lo pseudolistato è

simile si può semplificare la descrizione utilizzando la ripetizione. Nel secondo livello si ha:

```
Scegli il colore dello sfondo
Scegli il colore del bordo
Scegli il colore del carattere
```

che può essere modificato utilizzando la funzione REPEAT per evitare ridondanze di istruzioni. Si noti che il lavoro precedentemente fatto verrà in parte scartato, affrontando il problema in maniera diversa. A questo punto, non essendo stato speso troppo tempo, è facile ricominciare. Bisogna imparare a cambiare soluzione ogni volta che se ne intravvede una migliore. Non si deve aver timore di scartare quella vecchia. È un errore frequente abituarsi a risolvere i problemi in un solo modo, cercate di evitarlo.

```
Posizionati sul primo colore del tipo
REPEAT finché non siano stati inseriti tutti i colori
    Scegli questo colore di tipo
    Avanza sul colore successivo
ENDREPEAT
```

Questa soluzione è più semplice e più compatta di quella precedente.

16.3 LO PSEUDOLISTATO PER L'ISTRUZIONE SCEGLI

Disponiamo adesso di tutti i mezzi per rifinire l'istruzione Scegli il colore. Ci si può basare sull'istruzione Scegli il colore dello sfondo per affrontare l'istruzione Scegli questo colore tipo. Consideriamo ora tutti i livelli di istruzioni a partire da Scegli il colore dello sfondo per costruire una procedura completa per la scelta del colore.

```
Considera il primo colore per il tipo
REPEAT finché tutti i colori non siano stati inseriti
    REPEAT finché non sia stato introdotto un colore lecito
        Inserisci il colore per il tipo
        Supponi che il colore non sia stato trovato
        SELECT l'ingresso da tastiera
        (L'ingresso da tastiera è un numero)
            SELECT il numero nell'intervallo
            (Il numero non è nell'intervallo)
```

```
                Poni il numero del colore al valore illecito standard
            ENDSELECT
        (L'ingresso da tastiera è il nome per intero del colore)
            Posizionati sul primo colore della tabella
            REPEAT finché tutti i colori non siano stati controllati OR
                il colore non sia stato trovato
                SELECT sul colore inserito
                (Sì, il colore è nella tabella)
                    Fissa il numero del colore in base al nome per
                        intero
                    Il colore fissato è stato trovato
                ENDSELECT
                Avanza al colore successivo
            ENDREPEAT
        (L'ingresso da tastiera è il nome abbreviato del colore)
            Posizionati sul primo colore della tabella
            REPEAT finché tutti i colori non siano stati controllati OR
                il colore non sia stato trovato
                SELECT sul colore inserito
                (Sì, il colore è nella tabella)
                    Fissa il numero del colore in base alla
                        abbreviazione
                    Il colore fissato è stato trovato
                ENDSELECT
                Avanza al colore successivo
            ENDREPEAT
        ENDSELECT
        SELECT il colore lecito per questo tipo
        (valore illecito per questo tipo)
            Visualizza il messaggio di errore
            Visualizza i colori leciti per questo tipo
        (valore lecito per questo tipo)
            Salva questo colore lecito
        ENDSELECT
    ENDREPEAT
    Avanza al colore successivo
ENDREPEAT
```

16.4 PSEUDOLISTATO DEI CARATTERI VIC

Lo pseudolistato per i caratteri VIC è più comprensibile, poiché è già stata fatta la maggior parte del lavoro nell'ottenere i colori desiderati. Ora si può rifinire l'istruzione Disponi il VIC con quei colori. Vi è ancora da allestire il controllo del carattere "invisibile", descritto nella domanda 2. Il secondo livello dell'istruzione Prepara il VIC con quei colori è il seguente:

```
SELECT sulla combinazione dei colori
(La combinazione dei colori è lecita)
    Converti i colori nel formato VIC
    Fissa il VIC con i colori stabiliti
(OTHERWISE)
    Visualizza il messaggio di combinazione di colori illecita
ENDSELECT
```

I dati richiesti per una combinazione lecita del colore dello sfondo e di quello del bordo aggiungeranno altre due righe alla tabella dei colori. Questo programma si chiamerà Demo.

16.5 MISCELLANEA

Questo paragrafo contiene lo pseudolistato per l'inizializzazione dei dati, la visualizzazione del messaggio di errore e delle scelte effettuate.

1. *Inizializzazione* Questa sequenza di pseudolistato descrive le cose da fare all'inizio del programma.

```
Riserva dello spazio per i codici dei colori (3 byte)
    per i nomi interi dei colori
    e per quelli abbreviati
REPEAT finché tutti i colori non siano stati inseriti
    Prendi il nome per intero
    Prendi il nome abbreviato
    Considera il valore del primo colore
    REPEAT finché tutti i valori dei colori non siano stati inseriti
      Prendi il valore del colore
        ENDREPEAT
ENDREPEAT
Predisponi i messaggi per lo sfondo, per il bordo
    e per il carattere
```

2. *Visualizzazione d'errore* La visualizzazione d'errore evidenzia i dati errati e indica i nomi dei colori leciti che possono essere inseriti per il tipo dato.

```
Visualizza il nome errato del colore
Visualizza il messaggio
Considera il valore del primo colore
REPEAT finché tutti i valori dei colori non siano stati controllati
    SELECT il colore del tipo
    (il colore è lecito)
        Visualizza l'abbreviazione lecita del colore
    ENDSELECT
    Avanza sul colore successivo
ENDREPEAT
```

3. *Predisposizione del video* Verrà qui descritta la visualizzazione sullo schermo del VIC dei colori scelti. Questo messaggio darà modo sia di controllare il programma, che di sintonizzare al meglio il vostro schermo televisivo. A seconda del tipo di video adoperato, si noterà la corrispondenza o meno dei colori prescelti.

```
Visualizza DEMO
Considera il primo colore del tipo
REPEAT finché tutti i colori non siano stati visualizzati
    Visualizza il nome del colore e la scelta effettuata
ENDREPEAT
```

16.6 DESCRIZIONE DEI DATI

La rifinitura dei dati necessari deve essere contemporanea a quella delle procedure adottate. Al livello superiore si ha:

```
scelta del colore = colore sfondo + colore bordo + colore carattere
```

Si inizia con una prima suddivisione dei tre tipi di colore. Il primo da considerare è quello dello sfondo.

```
colore sfondo = [numero lecito di sfondo |
                 nome lecito di sfondo |
                 abbreviazione lecita di sfondo]
```

Ora si deve stabilire quali sono i numeri, i nomi e le abbreviazioni lecite

per lo sfondo. Dallo sviluppo dello pseudolistato si osserva che i colori del bordo e del carattere sono molto simili a quelli per lo sfondo. Precedentemente è stata sfruttata questa somiglianza riutilizzando parte dello pseudolistato con un REPEAT, anziché una semplice ripetizione delle istruzioni. Anche qui si può fare la stessa cosa. Le parti comuni sono il numero, il nome e l'abbreviazione del colore. Le differenze sono identificate da un flag che specifica la validità del colore per il particolara carattere tipografico. Questa informazione è utile per definire il successivo livello di rifinitura dei dati.

numero lecito sfondo = colore + sfondo ok
nome lecito sfondo = nome colore + sfondo ok
abbreviazione lecita sfondo = abbreviazione colore + sfondo ok

I colori del bordo e del carattere sono assegnati allo stesso modo.

numero lecito bordo = numero colore + bordo ok
nome lecito bordo = nome colore + bordo ok
abbreviazione lecita bordo = abbreviazione colore + bordo ok
numero lecito carattere = numero colore + carattere ok
abbreviazione lecita carattere = abbreviazione colore + carattere ok

I numeri leciti per i colori si trovano sulla tastiera del VIC. Essi corrispondono ai tasti numerici da 1 a 8. Il dizionario dati sarà perciò il seguente:

numero del colore = [1|2|3|4|5|6|7|8]

I numeri dei colori consentiti si trovano nel manuale d'uso del VIC.

nome del colore = [nero|bianco|rosso|azzurro|viola|verde|blu|giallo|
arancio|arancione chiaro|rosa|celeste|
viola chiaro|verde chiaro|blu intermedio|giallo chiaro]

Le abbreviazioni per i primi otto colori possono essere desunte dalla tastiera. Per gli altri colori è necessario inventare nuovi termini, con l'unica limitazione della lunghezza massima di tre caratteri. Se si vuole cambiare la lunghezza delle abbreviazioni dei colori, occorre modificare il programma. Le abbreviazioni dei colori sono:

abbreviazione del colore = [blk|wht|red|cyn|pur|grn|blu|yel|orn|
lor|pnk|lcy|lpu|lgr|lbl|lye]

Osservando attentamente l'ordine dei numeri, dei nomi e delle abbrevia-

zioni dei colori, si nota la corrispondenza tra i tre insiemi.
Il passo successivo è quello di specificare i valori dei colori. Ciò è necessario per consentire al VIC di predisporre i colori. Nel manuale del VIC vi è un capitolo dedicato alle istruzioni POKE necessarie per cambiare i colori. Questo programma dimostrativo utilizzerà le stesse POKE. Dalla tabella delle combinazioni dei colori per lo sfondo e per il bordo si ottengono i valori del colore dello sfondo. Essi sono:

valore colore sfondo: [0|16|32|48|64|80|96|112|128|144|160|
176|192|208|224|240]

Questi colori di sfondo sono elencati nello stesso ordine dei numeri, dei nomi e delle abbreviazioni per i colori.
I valori del colore del bordo possono essere desunti dalla tabella delle combinazioni dello sfondo e del bordo. È un insieme composto da solo 8 elementi, poiché sono ammessi solo i colori fondamentali. Poiché vi sono 16 possibili nomi ed abbreviazioni per i colori, assumiamo che ve ne siano altrettanti per il colore del bordo. Gli ultimi otto colori, ovviamente, non sono leciti. Quindi:

valore colore bordo = [8|9|10|11|12|13|14|15|
8<(flag d'errore)>8]

I valori per il colore del carattere sono riportati nella appendice J del manuale del VIC. Essi sono distribuiti nella tabella dei caratteri. È bene ordinarli secondo lo stesso criterio adottato per i nomi e le abbreviazioni.

valore colore carattere = [144|5|28|159|156|30|31|158|
8<(flag d'errore)>8]

I colori del carattere sono anch'essi ristretti agli 8 fondamentali. Già come è stato fatto per i colori del bordo, i valori del colore dei caratteri sono 16, dei quali gli ultimi 8 sono però illeciti.
Ora si hanno a disposizione tutti i dati necessari alla costruzione di una tabella completa dei colori. Questa tabella disporrà di 16 possibili scelte per i nomi e le abbreviazioni. La diciassettesima possibilità della tabella verrà utilizzata come standard del colore illecito. Il dizionario dati assumerà così la forma:

tabella colore = 16<possibile colore>16 + colore illecito

Ciascuna possibilità sarà composta da tutte le informazioni inerenti quel dato colore

possibile colore = nome colore + abbreviazione colore + valore sfondo +
+ valore bordo + valore carattere

Il colore del tipo è l'ultimo dato ancora da descrivere.

colore tipo = [sfondo|bordo|carattere]

16.7 STESURA DEL PROGRAMMA

Questo approccio alla stesura di un programma è basato sul fatto che è facile scrivere programmi brevi e semplici. Quello che è stato fatto finora è suddividere un programma di media complessità in vari programmi più semplici. Questo approccio è del tipo *divide et impera*!
Prima di iniziare con la stesura del programma, si suddivide in intervalli la numerazione delle linee del programma a seconda delle varie sezioni.

0100-0199	Inizializzazione
0200-0300	Ciclo REPEAT superiore
3000-3999	Subroutine di scelta del colore
4000-4999	Predisposizione dei colori sul VIC
5000-	Dati per l'inizializzazione

Le prime istruzioni di programma che si scriveranno sono per il ciclo principale di controllo. L'istruzione REPEAT, finché l'operatore non trova la combinazione soddisfacente, può essere simulata facilmente. Basta utilizzare un ciclo non condizionale. Si può sempre uscire da un programma battendo STOP e RESTORE. Se, però, si utilizza questo ciclo e lo STOP/RESTORE per terminarlo, i colori dello schermo saranno riportati a quelli standard. Questa è una caratteristica del tasto RESTORE. Perciò bisogna inserire un controllo per terminare. Bisogna inoltre inserire un REM per dare un nome al programma. Un listato completo del programma è riportato in appendice A.

```
200 INPUT"CONTINUI (S/N) ";A$
205 PRINT"♥"
210 IFLEFT$(A$,1)="N"THENSTOP
220 IFLEFT$(A$,1)<>"S"THEN200
250 GOTO200
```

Per mantenere un certo controllo, è possibile inserire la linea 230 PRINT"SI/OK". Controllate questa parte con sì, no o qualcos'altro.

Quindi aggiungete la parte relativa all'inizializzazione. Definite i dati necessari e inseriteli nel programma, controllandone il perfetto funzionamento. Lo pseudolistato per l'inizializzazione è:

```
Riserva spazio per i codici dei colori (3), per i nomi per esteso,
    per le abbreviazioni
REPEAT finché tutti i colori non siano inseriti
    Prendi il nome per esteso
    Prendi l'abbreviazione
    Parti con il primo valore del colore
    REPEAT finché tutti i valori dei colori siano inseriti
        Prendi il valore del colore
    ENDREPEAT
ENDREPEAT
Predisponi i messaggi per il bordo, per lo sfondo, per il carattere
```

Le istruzioni necessarie per terminare questa inizializzazione sono:

```
100 NC=17:DIMCO%(NC,3),FU$(NC),AB$(NC)
110 FORI=1TONC:READFU$(I),AB$(I)
120 FORJ=1TO3:READCO%(I,J)
130 NEXTJ
150 NEXTI
161 CC$(1)="LO SFONDO"
162 CC$(2)="IL BORDO"
163 CC$(3)="IL CURSORE"
```

Prima di effettuare il controllo di questa parte, bisogna inserire i dati per l'istruzione READ.

```
5001 DATANERO,BLK,0,8,144
5002 DATABIANCO,WHT,16,9,5
5003 DATAROSSO,RED,32,10,28
5004 DATAAZZURRO,CYN,48,11,159
5005 DATAVIOLA,PUR,64,12,156
5006 DATAVERDE,GRN,80,13,30
5007 DATABLU,BLU,96,14,31
5008 DATAGIALLO,YEL,112,15,158
5009 DATAARANCIO,ORN,128,-1,-1
5010 DATARANCIO CHIARO,LOR,144,-1,-1
5011 DATAROSA,PNK,160,-1,-1
5012 DATACELESTE,LCY,176,-1,-1
5013 DATAVIOLA CHIARO,LPU,192,-1,-1
5014 DATAVERDE CHIARO,LGR,208,-1,-1
5015 DATABLU INTERMEDIO,LBL,224,-1,-1
```

```
5016 DATAGIALLO CHIARO,LYE,240,-1,-1
5017 DATAILLECITO,ILL,-1,-1,-1
```

Vi sono varie tecniche interessanti in questa definizione dei dati. Si noti che il codice del colore è uguale al numero dell'istruzione meno 5000. Questi dati saranno utili per il controllo del programma. Quando il flag assume il valore -1, vuol dire che il colore scelto è illecito per il bordo o per il carattere. L'ultima linea del programma (5017) contiene un ingresso per i dati illeciti e tutti e tre i valori di dati vengono fissati al valore -1. Controllate questa porzione di programma aggiungendo le seguenti linee:

```
140 PRINT I;FU$(I);AB$(I);CO%(I,1);CO%(I,2);CO%(I,3)
145 GETA$:IFA$="" GOTO145
```

Questa aggiunta permetterà il controllo dei dati, visualizzando ciascuna scelta ed attendendo la battuta di un tasto per continuare. È possibile verificare ogni ingresso ed assicurarsi della sua corretta sintassi. Si faccia partire il programma con l'istruzione RUN.
Supponiamo che durante il controllo le ultime due linee siano:

```
6 VERDEGRN 80  13  30
?TYPE MISMATCH
 ERROR IN 110
```

Ciò indica che occorre immediatamente listare la linea 5007 che sarebbe stata letta immediatamente dopo. Se vi è un errore nei dati, si può risalire alla linea interessata aggiungendo 5000 al codice del colore. Dopo avere controllato che tutti i dati sono esatti, eliminate le linee 140 e 145. Il successivo passo da eseguire è quello di inserire le linee di programma per l'ingresso numerico dei colori. Innanzitutto aggiungete una chiamata alla subroutine nel ciclo di controllo principale, quindi introducete la subroutine stessa.

```
230 GOSUB 3000
3000 FORCC=1TO3
3100 PRINT"COLORE PER ";CC$(CC)
3110 INPUTA$
3120 IFLEN(A$)<3THENCN=VAL(A$):GOTO3200
3200 IFCN<1 OR CN>NC-1 THENCN=NC:GOTO3300
3210 IFCO%(CN,CC)=-1THENCN=NC
3300 IFCN<>NCTHEN3400
3310 PRINTA$;" NON VA BENE PER ";CC$(CC)
3350 GOTO3100
```

```
3400 CV(CC)=CN
3410 NEXTCC
3420 RETURN
```

Controllate l'efficienza della subroutine con numeri di colori leciti ed illeciti. Noterete che i numeri da 9 a 16 sono leciti per quei colori del bordo non indicati sulla tastiera. Notate che il messaggio di errore è insufficiente. Aggiungete le seguenti linee per visualizzare il listato dei colori leciti in caso di errore.

```
3320 FORI=1TONC
3325 IFCO%(I,CC)<>-1THENPRINTAB$(I);",";
3330 NEXTI
3340 PRINT"▌ VANNO BENE"
```

Il cursore a sinistra della linea 3340 cancella l'ultima virgola. Controllate nuovamente i numeri dei colori illeciti. Si dovrebbe ottenere un elenco di colori validi per un carattere tipografico, mediante il messaggio di errore. Per assicurarvi che siano stati scelti i colori giusti, aggiungete le seguenti linee per visualizzare quanto inserito.

```
240 GOSUB 4000
4000 IFCV(1)<>CV(3)THEN4030
4010 PRINT"NON VA BENE- NON PUOI AVERE SIA SFONDO CHE  CURSORE ";
     FU$(CV(1))
4020 RETURN
4030 REM
4060 PRINTTAB(9);"PROVA"
4070 FORI=1TO3
4080 PRINTCC$(I);" ";FU$(CV(I))
4090 NEXTI
4099 RETURN
```

Adesso controllate l'input numerico. Esiste una corrispondenza tra i codici dei colori ed i nomi dei colori visualizzati? Se ciò non fosse, controllate le istruzioni DATA con numero di linea corrispondente al codice del colore più 5000. È opportuno controllare anche il colore dello sfondo ed il colore del carattere. La successiva parte da aggiungere riguarda i nomi abbreviati.

```
3140 GOSUB 3600
3600 FORCN=1TONC
3610 IFA$=AB$(CN)THENRETURN
3620 NEXTCN
3630 CN=NC
3640 RETURN
```

Controllate questa parte con abbreviazioni lecite ed illecite. Il passo finale è quello di aggiungere le istruzioni per accettare i nomi per esteso dei colori in quella parte di programma riguardante la scelta dei colori.

```
3130 IF LEN(A$)>3 THEN GOSUB3500:GOTO3200
3500 FORCN=1TONC
3510 IFA$=FU$(CN)THENRETURN
3520 NEXTCN
3530 CN=NC
3540 RETURN
```

Controllate questa parte con nomi leciti ed illeciti. Sarebbe opportuno controllare nuovamente l'ingresso numerico e l'ingresso dei nomi abbreviati per assicurarsi che le nuove istruzioni non abbiano cancellato quelle precedenti. Provate un colore come AZZURRO per il colore del bordo o per quello del carattere.
Il passo finale è quello di aggiungere le istruzioni necessarie a predisporre i colori scelti sul VIC. Occorre fare attenzione a non usare indirizzi errati con l'istruzione POKE, poiché si corre il rischio di alterare delle locazioni riservate al sistema operativo della macchina. Se si commette tale errore, è sufficiente premere STOP e RESTORE per sbloccare il computer. Se temete di perdere il programma costato tanta fatica, è sufficiente farne una copia su cassetta, dopo l'introduzione delle seguenti tre linee:

```
4030 POKE36879,CO%(CV(1),1)+CO%(CV(2),2)
4040 C$=CHR$(CO%(CV(3),3))
4050 PRINTC$
```

Questo (finalmente!) completa il programma dimostrativo sulle capacità grafiche del VIC. Controllatelo con varie combinazioni numeriche e non. Quali sono le combinazioni più gradevoli?

CAPITOLO DICIASSETTE

Suono

Il VIC è dotato di un generatore audio che può simulare, attraverso l'altoparlante del televisore, suoni e musica. Questo capitolo utilizzerà le tecniche introdotte nel capitolo 15 per la stesura di un semplice programma di composizione musicale. Queste tecniche sono state utilizzate nel capitolo 16 con lo sviluppo del programma dimostrativo sulle capacità grafiche a colori del VIC. Se non avete letto i precedenti capitoli 15 e 16, è opportuno farlo ora. Così facendo, questo capitolo sarà più facile da seguire.

17.1 IL PROBLEMA

Vogliamo sviluppare un programma che utilizzi le caratteristiche audio del VIC. Questo programma deve permettere l'inserimento, l'ascolto, la memorizzazione ed il richiamo di una melodia da una cassetta. I dati necessari per specificare una melodia sono l'ottava, la nota e la durata di quest'ultima. Quando una melodia viene memorizzata, occorre assegnarle un nome per identificarla. Quest'ultimo sarà utilizzato successivamente per richiamare la melodia dal registratore. L'ascolto di una melodia si basa sul susseguirsi delle note dalla prima all'ultima, fino al termine della melodia stessa. Le note sono indicate nella maniera classica (DO, RE, RE # , ecc.) e spetta al programma la codificazione in dati comprensibili dal VIC.

17.2 RESTRIZIONI

Il programma di composizione musicale dimostra le capacità audio del VIC. Vi è bisogno di qualche semplificazione al programma dimostrativo che faciliti il suo svolgimento. Questo programma non ha la pretesa di sostituirsi al compositore che ricerca nuove melodie.
Il programma ha i seguenti limiti:

1. Sono permesse solo singole note e non è quindi possibile eseguire accordi musicali.
2. Ogni melodia verrà suonata dall'inizio alla fine. Non esiste un metodo per ascoltarne una parte.
3. Se la melodia ha un accompagnamento, dovrà essere inserito dopo ogni verso.
4. Una volta inserita una melodia, l'unico modo per cambiarla è reinserire la melodia per intero. Non esiste un modo per correggere o cambiare una melodia già inserita.

Sviluppando successivamente il programma, è possibile eliminare qualcuna di queste restrizioni.

17.3 SVILUPPO DEL PROGRAMMA

Con le specifiche indicate in 17.1 e le restrizioni indicate in 17.2, usando gli strumenti del capitolo 15, è possibile sviluppare il programma audio al primo livello.

```
Inizializza i dati
REPEAT finché non si richieda un'altra melodia
    Scegli l'azione voluta
    SELECT l'azione
    (l'azione è leggere)
        Leggi la melodia dal nastro
    (l'azione è registrare)
        Registra la melodia sul nastro
    (l'azione è comporre)
      Componi la melodia
      (l'azione è riprodurre)
        Esegui la melodia
      (OTHERWISE)
        Visualizza la scelta errata
    ENDSELECT
ENDREPEAT
```

Questo definisce il problema in termini di pseudolistato. Il passo successivo è quello di sviluppare le istruzioni generali del primo livello per la descrizione dettagliata di ciò che si richiede.

17.4 RIFINITURA DELLA SOLUZIONE

Si inizia a perfezionare la soluzione con le quattro istruzioni di azione. Perciò Leggi la melodia dal nastro diventerà:

```
Scegli il nome della melodia
OPEN l'archivio su nastro per leggere la melodia
Inizia con la prima nota
REPEAT finché tutte le note non siano state lette
    Leggi il valore della nota e la durata (dal nastro)
    Salva il valore e la durata nella melodia
    Passa alla nota successiva
ENDREPEAT
CLOSE l'archivio su nastro
```

Lo sviluppo dell'istruzione Registra la melodia sul nastro è molto simile, ad eccezione della direzione del flusso di dati:

```
Scegli il nome della melodia
OPEN l'archivio su nastro per registrare la melodia
Inizia con la prima nota
REPEAT finché tutte le note non siano state registrate
    Registra il valore della nota e la durata
    Passa alla nota successiva
ENDREPEAT
Registra l'indicatore di fine melodia
CLOSE l'archivio su nastro
```

L'istruzione Componi la melodia viene sviluppata in modo analogo:

```
Inizia con la prima nota
REPEAT finché tutte le note non siano state inserite
    Prendi l'ottava, la nota e la durata dalla tastiera
    Converti l'ottava e la nota nel valore VIC corrispondente
    Salva il valore e la durata della nota
    Passa alla nota successiva
ENDREPEAT
```

L'ultima delle quattro azioni è Esegui la melodia. Può essere sviluppata nel modo seguente:

```
Inizia con la prima nota
REPEAT finché tutte le note non siano state eseguite
    Prendi il valore della nota e della durata dalla melodia
    Riproduci la nota per la sua durata
ENDREPEAT
```

I dati richiesti per queste quattro azioni sono l'ottava, la nota, la durata, l'indicatore di fine melodia, la melodia (insieme di note) e la rappresentazione del VIC per le ottave e le note (valore delle note).
Si può sviluppare ulteriormente l'istruzione Prendi l'ottava, la nota e la durata dalla tastiera dello pseudolistato Componi la melodia. Occorre decidere riguardo alla richiesta di ingresso ed al suo formato. Lo sviluppo di Prendi l'ottava sarà:

```
REPEAT finché non sia stata inserita l'ottava lecita
    Richiedi l'ottava
    Scegli l'ottava dalla tastiera
    Inizia con il primo nome dell'ottava
    REPEAT finché non siano stati controllati tutti i nomi di ottava
            OR finché non si trovi il nome dell'ottava
        SELECT il nome di ottava inserito
        (coincide con il nome di ottava della tabella)
            Fissa il nome di ottava trovato
            Salva l'ottava
        ENDSELECT
        Passa al nome di ottava successivo
    ENDREPEAT
ENDREPEAT
```

Lo sviluppo di Scegli la nota viene eseguito in modo analogo:

```
REPEAT finché non sia stata inserita la nota lecita
    Richiedi la nota
    Scegli la nota dalla tastiera
    REPEAT finché non siano stati controllati tutti i nomi di nota
        OR finché non si trovi il nome della nota
    SELECT il nome di nota inserito
    (coincide con il nome di nota della tabella)
        Fissa il nome di nota trovato
        Salva la nota
    ENDSELECT
```

```
      Passa al nome di nota successivo
    ENDREPEAT
  ENDREPEAT
```

La durata viene specificata nello stesso modo. Lo sviluppo di Scegli la durata sarà:

```
REPEAT finché non sia stata inserita la durata lecita
    Richiedi la durata
    Scegli la durata dalla tastiera
    REPEAT finché non siano state controllate tutte le durate
            OR finché non si trovi la durata lecita
        SELECT la durata
        (coincide con la durata in tabella)
            Fissa la durata trovata
            Salva la durata
        ENDSELECT
        Passa alla durata successiva
    ENDREPEAT
ENDREPEAT
```

Notate la similitudine con lo sviluppo del programma dimostrativo sulle capacità grafiche del VIC del capitolo 16. In quel programma, i passi ripetitivi sono stati sostituiti da un unico ciclo generico di REPEAT. In questo caso non verrà effettuata tale generalizzazione. Ciò indica che esistono vari modi per creare un programma. Un esercizio utile per il futuro potrà essere quello di convertire i vari cicli in un unico ciclo REPEAT per i vari tipi di ingressi di note. Provate ciò dopo avere compreso correttamente il funzionamento di questa versione di programma.
L'ultima istruzione di pseudolistato da ampliare è Esegui la melodia. L'istruzione Riproduci la nota per la sua durata diverrà:

```
POKE il valore della nota nel generatore 2
POKE degli zero negli altri generatori
Attiva il volume
Memorizza il tempo
REPEAT finché non è passato un tempo corrispondente alla durata
ENDREPEAT
Disattiva il volume
```

17.5 DEFINIZIONE DEI DATI

I dati necessari sono i vari tipi di azioni necessarie, l'ottava, la nota, la durata, la melodia e i valori delle note nel VIC. Il primo dizionario dati è quello delle azioni.

azione = [leggere|memorizzare|comporre|riprodurre]

Definiamo ora l'ottava. Ve ne sono tre possibili per ogni generatore audio: alta, media e bassa. Usando una lettera per indicare l'ottava, il dizionario dati per le ottave sarà:

ottava = [a|m|b]

Il manuale VIC elenca le 12 note che possono essere definite:

nota = [DO|DO#|RE|RE#|MI|FA|FA#|SOL|SOL#|LA|LA#|SI]

Questi sono i nomi delle note. È possibile aggiungere p come pausa per una nota nulla (silenzio). Una buona estensione di questo programma potrebbe essere quella di aggiungere le note in bemolle. Provate dopo avere compreso il funzionamento del programma di base. I dati successivi da definire sono le durate delle note. Si considerino solo le durate semplici di 1/16, 1/8, 1/4, 1/2 e dell'intera nota. Indicando con singoli caratteri queste durate, si otterrà:

durata = [s|8|4|2|1]

La melodia può essere rappresentata da un insieme di valori di note e durate. A causa delle limitazioni di memoria a disposizione, occorre stabilire un limite superiore per la lunghezza della melodia. Occorre anche riservare spazio per gli accordi con gli altri generatori. In questa versione del programma, gli altri due generatori rimarranno spenti. Il dizionario dati per la melodia è:

melodia = 0<valore nota+spazio+durata>ms

Nell'ingresso del dizionario dati, ms rappresenta il numero di note della melodia più lunga. L'ultimo dizionario dati da definire è la rappresentazione del VIC per ognuna delle note possibili. Poiché la nota e l'ottava sono state già definite, si organizzerà la tabella in tre colonne (una per ciascuna ottava) di dodici righe ciascuna (una per ogni nota). Una descrizione generale del valore di una nota sarà:

valore nota = 3<12<POKE # > 12>3

Il POKE # si può trovare nel manuale del VIC. La prima nota definita è stata il DO. I valori di POKE per la nota DO sono 135, 195 e 225. Questi valori sono in ordine crescente di ottava come definito in precedenza: bassa, media, alta. Considerando anche i valori sul manuale VIC delle restanti note si avrà:

POKE # =	[135	195	225
	143	199	227
	147	201	228
	151	203	229
	159	207	231
	163	209	232
	167	212	233
	175	215	235
	179	217	236
	183	219	237
	187	221	238
	191	223	239]

Con questa tabella terminano i dati principali necessari al programma dimostrativo delle capacità musicali del VIC.

17.6 STESURA DEL PROGRAMMA

Con tutti gli pseudolistati compilati precedentemente, è possibile numerare a blocchi le istruzioni delle varie parti del programma. Un listato completo del programma audio è riportato in appendice A. È opportuno salvare su nastro le varie parti del programma non appena vengono inserite. Questo è bene farlo dopo aver inserito ciascuna parte principale. In questo modo, se per disgrazia si battesse NEW o mancasse l'energia elettrica, non si perderebbe molto tempo. Per la numerazione delle istruzioni si adotti la seguente convenzione:

0100-0199	Inizializzazione
0200-0499	Ciclo REPEAT superiore
1100-1199	Lettura melodia
1200-1299	Memorizzazione melodia
1300-1399	Composizione melodia
1400-1499	Riproduzione melodia
3000-3999	Richiesta di input
4000-4999	Input da tastiera
5000-	Dati per l'inizializzazione

Le prime istruzioni da usare sono quelle del ciclo principale di controllo. Il REPEAT finché non si richieda un'altra melodia può essere eseguito molto facilmente. È sufficiente un ciclo incondizionato. È sempre possibile terminare il programma mediante i tasti STOP e RESTORE. Questo programma differisce dal programma sul colore poiché non si desidera conservare lo stato finale del programma. Inserite anche un commento per sapere in futuro di che programma si tratta.

```
200 PRINT"♥"
410 GOTO 200
```

Le successive istruzioni da introdurre sono quelle relative all'inizializzazione. È necessario preservare dello spazio per la melodia, per i nomi delle note, delle ottave e delle durate, e per tutti i valori delle note. Inoltre, è necessario definire il numero delle azioni, ciascuno degli indirizzi dei generatori audio del VIC ed il volume.

```
100 MS=40: NN=12: NO=3: DIM SO%(MS,4),OC$(NO),NO$(NN)
102 NL=5: DIM NL$(NL),NL(NL),SC%(NN,NO)
105 NA=4: FOR I=1 TO 5: SI=36873+I: NEXT I: VO=12
```

Occorre adesso caricare i nomi ed i valori delle 12 note per ciascuna ottava. La procedura si divide in due parti. La prima è costituita dalle istruzioni READ e la seconda dalle istruzioni DATA.

```
110 FOR I=0 TO NN: READ NO$(I)
120 FOR J=1 TO NO: READ SC%(I,J)
130 NEXT J
150 NEXT I
5000 DATA R,0,0,0
5001 DATA DO,135,195,225
5002 DATA DO#,143,199,227
5003 DATA RE,147,201,228
5004 DATA RE#,151,203,229
5005 DATA MI,159,207,231
5006 DATA FA,163,209,232
5007 DATA FA#,167,212,233
5008 DATA SOL,175,215,235
5009 DATA SOL#,179,217,236
5010 DATA LA,183,219,237
5011 DATA LA#,187,221,238
5012 DATA SI,191,223,239
```

Per controllare questa parte di programma ed essere sicuri di aver intro-

dotto correttamente i dati, si aggiungano queste due linee.

```
140 PRINT I;NO$(I);SC%(I,1);SC%(I,2);SC%(I,3)
145 GET A$: IF A$="" GOTO 145
```

Dopo aver effettuato questo controllo, si cancellino le linee 140 e 145. Si inseriscano ora i nomi delle tre parti costituenti la nota (ottava, nome e durata), le quattro azioni (lettura, memorizzazione, composizione e riproduzione) e le tre ottave (alta, media e bassa).

```
161 CC$(1)="OTTAVA"
162 CC$(2)="NOTA"
163 CC$(3)="DURATA"
171 AC$(1)="LEGGERE"
172 AC$(2)="SCRIVERE"
173 AC$(3)="COMPORRE"
174 AC$(4)="SUONARE"
181 OC$(1)="B"
182 OC$(2)="M"
183 OC$(3)="A"
```

Le successive informazioni da inserire sono i nomi delle differenti durate delle note ed i loro rispettivi valori. Le durate sono 2 secondi per l'intera nota e 1/30 di secondo (NL(0)) come stacco tra una nota e l'altra. Volendo cambiare la cadenza delle note, occorre modificare le seguenti istruzioni.

```
190 NL$(0)="": NL(0)=(1/30)*60
191 NL$(1)="1": NL(1)=(1/16)*120
192 NL$(2)="2": NL(2)=(1/8)*120
193 NL$(3)="4": NL(3)=(1/4)*120
194 NL$(4)="8": NL(4)=(1/2)*120
195 NL$(5)="16": NL(5)=1*120
```

Ciò completa l'inizializzazione del programma dimostrativo sulle capacità musicali del VIC. Il passo successivo è quello di aggiungere le istruzioni per Scegli l'azione voluta e per SELECT l'azione.

```
300 PRINT"COSA VUOI FARE"
310 INPUT A$
320 FOR I=1 TO NA
330 IF AC$(I)=A$ THEN 400
340 NEXT I
350 PRINT"PUOI SCEGLIERE FRA:"
360 FOR I=1 TO NA: PRINT AC$(I): NEXT I
```

```
370 GOTO 300
400 ON I GOSUB 1100,1200,1300,1400
```

Questa parte di programma controlla la tabella delle azioni lecite. Se un'azione non viene trovata, essa visualizza le azioni consentite e chiede di riprovare. Se invece viene trovata un'azione, il controllo passa ad una subroutine della serie 1N00, ove N è il numero dell'azione scelta. Utilizzando numeri di linea che rispecchiano l'organizzazione del programma, è più semplice effettuare la caccia agli errori nel programma stesso. Cercate di utilizzare delle vostre tecniche per l'organizzazione del programma. La parte successiva del programma da inserire è l'istruzione Componi la melodia. Il ciclo più esterno è quello del REPEAT finché tutte le note non siano state inserite.

```
1300 IP=0
1350 GOSUB 3000
1360 IF OK=1 THEN RETURN
1370 IF IP<MS THEN 1350
1380 PRINT"FATTO. NON C'E' PIU' SPAZIO": RETURN
```

Questa parte di programma carica le note finché non vi è più spazio in memoria o non vi sono più note. Il lavoro di caricamento delle note è eseguito dalla subroutine 3000 che verrà inserita più tardi. Si inizi ora con Scegli l'ottava.

```
3000 CC=1: OK=0: PRINT"] NOTA";IP+1
3010 GOSUB 4000: IF A$="FINE" THEN OK=1: RETURN
3020 IP=IP+1
3040 FOR CN=1 TO NO
3050 IF A$=OC$(CN) THEN 3100
3060 NEXT CN
3070 PRINT"ERRORE. LA ";CC$(CC): PRINT"PUO' ESSERE:"
3080 FOR I=1 TO NO: PRINT OC$(I): NEXT I
3090 IP=IP-1: GOTO 3010
```

Questa parte di programma utilizza la subroutine 4000 per richiedere ed ottenere il nome dell'ottava come dato di ingresso. Per controllare questa parte di programma, si aggiunga la subroutine 4000 e le due linee di programma per completare la subroutine 3000.

```
4000 PRINT"BATTI LA ";CC$(CC)
4010 INPUT A$: RETURN
3100 REM
3999 RETURN
```

Controllate questa parte di programma. Ricordate che, per ora, si può solo comporre, mentre, scegliendo altre azioni, si ottengono errori del tipo UNDEFINED STATEMENT NUMBER ERROR. Dopo il controllo dell'istruzione Scegli l'ottava, si possono inserire le istruzioni Scegli la nota. Esse sono:

```
3100 CC=2: GOSUB 4000
3110 FOR I=0 TO NN: IF A$=NO$(I) THEN 3200
3120 NEXT I
3130 PRINT"NON ESISTE. ESISTONO:";
3140 FOR I=1 TO NN: PRINT NO$(I);" ";: NEXT I: PRINT
3150 GOTO 3100
```

La melodia si compone di quattro parti per nota. Le variabili SO%(NOTA,1) e SO%(NOTA,3) sono le pause necessarie agli accordi. In questa versione di programma saranno entrambe nulle. La variabile SO%(NOTA,2) è la singola nota da riprodurre e la variabile SO%(NOTA,4) è la durata di quella nota. Le istruzioni dello pseudolistato Converti l'ottava e la nota nel valore VIC corrispondenti e Salva il valore della nota nella melodia possono essere eseguite in un'unica istruzione Basic.

```
3200 SO%(IP,2)=SC%(I,CN)
```

Quando viene eseguita la linea 3200, la variabile I e la variabile CN rappresentano rispettivamente il numero della nota ed il numero dell'ottava. Con questi dati è sufficiente ricercare il valore della nota nella tabella inizializzata alla linea 120. Convertire la nota nel corrispondente valore VIC è estremamente facile (con una giusta scelta per la rappresentazione dei dati).
Ora è possibile estendere il controllo inserendo sia l'ottava che la nota. Si provino sia valori leciti che non, per la nota e l'ottava. I successivi passi da inserire nel programma sono Scegli la durata e Salva la durata nella melodia. Essi sono:

```
3210 CC=3: GOSUB 4000
3220 FOR I=1 TO NL: IF A$=NL$(I) THEN 3300
3230 NEXT I
3240 PRINT"NON ESISTE. ESISTONO:";
3250 FOR I=1 TO NL: PRINT NL$(I);" ";: NEXT I: PRINT
3260 GOTO 3210
3300 SO%(IP,4)=NL(I)
```

Per consentire un ulteriore sviluppo in futuro, è stato riservato dello spazio nella melodia per altre due note. Questo permetterà di aggiungere gli accordi successivamente. Assicuratevi che venga inserita una sola no-

ta in questa versione di programma dimostrativo sull'audio del VIC, ponendo a zero le altre note. Ciò si può fare con la sola linea:

```
2310 SO%(IP,1)=0: SO%(IP,3)=0
```

Questo completa la parte di programma Componi la melodia. Si inseriscano ora le istruzioni per Riproduci la melodia.

```
1400 IF IP=0 THEN RETURN
1410 FOR I=1 TO IP: FOR J=1 TO 3: POKE S(J),SO%(I,J)
1420 NEXT J
1425 POKE S(5),VO
```

Questo dispone ognuno dei tre generatori audio ai valori delle note salvate nella melodia ed inoltre inserisce il volume. Ora bisogna cronometrare il tempo di durata della nota. Si ottiene ciò con:

```
1430 X=TI+SO%(I,4)
1440 IF TI<X THEN 1440
1450 POKE S(5),0
1460 X=TI+NL(0)
1470 IF TI<X THEN 1470
1480 NEXT I
1490 RETURN
```

La linea 1430 calcola quando termina il suono della nota. La linea 1450 disinserisce il volume. Una breve pausa (1/30 di secondo) separa le note con le linee 1460 e 1470.
Questo completa la parte del programma Riproduci la melodia. Controllate il tutto inserendo una melodia conosciuta. Ricordate che bisogna ancora inserire le istruzioni di programma che memorizzano la melodia su nastro. Ciò significa che è opportuno scegliere una breve melodia per il controllo. Dopo aver controllato le azioni di composizione e riproduzione, si può aggiungere la parte di programma Registra la melodia su nastro:

```
1200 INPUT"NOME DELLA CANZONE";A$
1210 OPEN 1,1,1,A$
1215 IF IP=0 THEN 1230
1220 FOR I=1 TO IP: FOR J=1 TO 4:
     PRINT# 1,SO%(I,J): NEXT J,I
1230 PRINT# 1,-1: CLOSE1
1240 RETURN
```

La linea 1230 memorizza sul nastro il segnale −1 di fine melodia. Fate attenzione a non usare un nastro contenente programmi o dati di valore.

Il successivo e ultimo passo è di inserire la subroutine dell'azione Leggi la melodia dal nastro.

```
1100 INPUT"NOME DELLA CANZONE";A$
1110 OPEN 1,1,0,A$
1120 IP=0
1130 FOR I=1 TO 4: INPUT# 1,X
1140 IF X<0 THEN 1180
1150 SO%(IP+1,I)=X: NEXT I
1160 IP=IP+1: IF IP<MS THEN 1130
1170 PRINT"CANZONE TROPPO LUNGA"
1180 CLOSE1: RETURN
```

La linea 1160 è importante poiché assicura che la melodia letta entri nello spazio riservato ad essa. È buona norma controllare che il dato sia contenuto nell'intervallo lecito. Questo completa il programma dimostrativo delle capacità musicali del VIC. Dopo aver verificato il funzionamento, memorizzate la versione finale su nastro. È possibile ora comporre e riprodurre melodie a volontà. Tuttavia, da buoni programmatori, si avranno certamente mille idee per migliorare questo programma.

Appendice A

LISTATO COMPLETO, IN VERSIONE FINALE, DEL PROGRAMMA INDIRIZZARIO 4.

```
100 PRINT "MENU MAILING LIST"
101 AT$(1)="AGGIUNGERE"
102 AT$(2)="MODIFICARE"
103 AT$(3)="VISUALIZZARE"
104 AT$(4)="CONTARE"
110 INPUT"COSA VUOI FARE";A$
112 IF A$="" THEN STOP
115 FOR A=1 TO 4
120 IF A$<>AT$(A) THEN 135
130 ON A GOTO 1000,2000,3000,4000
135 NEXT A
140 PRINT "INPUT SBAGLIATO!"
150 GOTO 110
1000 PRINT "AGGIUNTA DI UN NOME"
1999 GOTO 150
2000 PRINT "CAMBIAMENTO DI UN NOME"
2990 GOTO 150
3000 PRINT "LISTA DEI NOMI"
3990 GOTO 150
4000 PRINT "CONTO DEI NOMI"
4990 GOTO 150
```

PROGRAMMA PER CONVERTIRE E VISUALIZZARE UN NUMERO DECIMALE IN UNA STRINGA BINARIA (DEC-BIN)

```
90 REM SCRIVE IL NUMERO BINARIO
100 INPUT "SCRIVI IL N. DECIMALE";R
430 R$=""
440 GOSUB 2000
500 PRINT "DECIMALE";R;"="
510 PRINT "BINARIO";R$
520 PRINT
530 GOTO 100
2000 REM CONVERSIONE DEL NUMERO IN STRINGA BINARIA
2010 C$(0)="0":C$(1)="1"
2020 R2=R
2030 T2=INT(R2/2)
2040 S=R2-T2*2
2050 R$=C$(S)+R$
2060 R2=T2
2065 IF LEN(R$)>16 THEN R$=RIGHT$(R$,16): RETURN
2070 IF T2<>0 THEN 2030
2080 RETURN
```

PROGRAMMA PER CONVERTIRE E VISUALIZZARE UNA STRINGA BINARIA IN UN NUMERO DECIMALE (BIN-DEC)

```
90 REM SCRIVE IL NUMERO DECIMALE
100 PRINT "SCRIVI IL N. BINARIO"
105 INPUT N$
106 IF N$="" THEN STOP
110 T1$=N$
120 GOSUB 1100
130 IF T<0 THEN 100
140 PRINT N$;" BINARIO"
145 PRINT "=";T;"DECIMALE"
150 GOTO 100
430 R$=""
440 GOSUB 2000
500 PRINT "DECIMALE";R;"="
510 PRINT "BINARIO";R$
520 PRINT
530 GOTO 100
1100 REM CONVERSIONE BINARIO-DECIMALE
1105 T=0
```

```
1110 L1=LEN(T1$)
1120 FOR I1=1 TO L1
1125 T=T*2
1130 IF MID$(T1$,I1,1)="1" THEN T=T+1: GOTO 1150
1140 IF MID$(T1$,I1,1)<>"0" THEN T=-1: RETURN
1150 NEXT I1
1160 RETURN
2000 REM CONVERSIONE DEL NUMERO IN STRINGA BINARIA
2010 C$(0)="0":C$(1)="1"
2020 R2=R
2030 T2=INT(R2/2)
2040 S=R2-T2*2
2050 R$=C$(S)+R$
2060 R2=T2
2065 IF LEN(R$)>16 THEN R$=RIGHT$(R$,16): RETURN
2070 IF T2<>0 THEN 2030
2080 RETURN
```

LISTATO COMPLETO DEL PROGRAMMA PER VISUALIZZARE FUNZIONI LOGICHE

```
90 REM FUNZIONI LOGICHE
100 PRINT "SCRIVI IL N. BINARIO"
105 INPUT N$
106 IF N$="" THEN STOP
110 T1$=N$
120 GOSUB 1100
130 IF T<0 THEN 100
140 NE=T
150 PRINT"SCRIVI LA FUN. LOGICA"
160 INPUT"(A,O,N)";OP$
190 F$=LEFT$(OP$,1)
200 IF F$<>"A" AND F$<>"O" AND F$<>"N" THEN 150
210 TW$=""
220 IF F$="N" THEN 420
300 PRINT"SCRIVI IL SECONDO NUM BINARIO"
305 INPUT TW$
310 T1$=TW$
320 GOSUB 1100
330 IF T<0 THEN 300
340 TWO=T
400 IF F$="A" THEN R=NE AND TWO
410 IF F$="O" THEN R=NE OR TWO
420 IF F$="N" THEN R=NOT NE
```

```
430 R$=""
440 GOSUB 2000
500 IF F$="N" THEN PRINT OP$;" ";N$;" ="
505 IF F$<>"N" THEN PRINT N$;" ";OP$;" ";TW$;" ="
510 PRINT R$
520 PRINT
530 GOTO 100
1100 REM CONVERSIONE BINARIO-DECIMALE
1105 T=0
1110 L1=LEN(T1$)
1120 FOR I1=1 TO L1
1125 T=T*2
1130 IF MID$(T1$,I1,1)="1" THEN T=T+1: GOTO 1150
1140 IF MID$(T1$,I1,1)<>"0" THEN T=-1: RETURN
1150 NEXT I1
1160 RETURN
2000 REM CONVERSIONE DEL NUMERO IN STRINGA BINARIA
2010 C$(0)="0":C$(1)="1"
2020 R2=R
2030 T2=INT(R2/2)
2040 S=R2-T2*2
2050 R$=C$(S)+R$
2060 R2=T2
2065 IF LEN(R$)>16 THEN R$=RIGHT$(R$,16): RETURN
2070 IF T2<>0 THEN 2030
2080 RETURN
```

LISTATO COMPLETO DEL PROGRAMMA DIMOSTRATIVO DELLE CAPACITÀ GRAFICHE DEL VIC

```
100 NC=17:DIMCO%(NC,3),FU$(NC),AB$(NC)
110 FORI=1TONC:READFU$(I),AB$(I)
120 FORJ=1TO3:READCO%(I,J)
130 NEXTJ
150 NEXTI
161 CC$(1)="LO SFONDO"
162 CC$(2)="IL BORDO"
163 CC$(3)="IL CURSORE"
200 INPUT"CONTINUI (S/N) ";A$
205 PRINT"♥"
210 IFLEFT$(A$,1)="N"THENSTOP
220 IFLEFT$(A$,1)<>"S"THEN200
230 GOSUB 3000
240 GOSUB4000
```

```
250 GOTO200
3000 FORCC=1TO3
3100 PRINT"COLORE PER ";CC$(CC)
3110 INPUTA$
3120 IFLEN(A$)<3THENCN=VAL(A$):GOTO3200
3130 IFLEN(A$)>3THENGOSUB3500:GOTO3200
3140 GOSUB3600
3200 IFCN<1 ORCN>NC-1 THENCN=NC:GOTO3300
3210 IFCO%(CN,CC)=-1THENCN=NC
3300 IFCN<>NCTHEN3400
3310 PRINTA$;" NON VA BENE PER ";CC$(CC)
3320 FORI=1TONC
3325 IFCO%(I,CC)<>-1THENPRINTAB$(I);",";
3330 NEXTI
3340 PRINT" I VANNO BENE"
3350 GOTO3100
3400 CV(CC)=CN
3410 NEXTCC
3420 RETURN
3500 FORCN=1TONC
3510 IFA$=FU$(CN)THENRETURN
3520 NEXTCN
3530 CN=NC
3540 RETURN
3600 FORCN=1TONC
3610 IFA$=AB$(CN)THENRETURN
3620 NEXTCN
3630 CN=NC
4000 IFCV(1)<>CV(3)THEN4030
4010 PRINT"NON VA BENE- NON PUOI AVERE SIA SFONDO CHE
     CURSORE";FU$(CV(1))
4020 RETURN
4030 POKE36879,CO%(CV(1),1)+CO%(CV(2),2)
4040 C$=CHR$(CO%(CV(3),3))
4050 PRINTC$
4060 PRINTTAB(9);"PROVA"
4070 FORI=1TO3
4080 PRINTCC$(I);" ";FU$(CV(I))
4090 NEXTI
4099 RETURN
5001 DATANERO,LLK,0,8,144
5002 DATABIANCO,WHT,16,9,5
5003 DATAROSSO,RED,32,10,28
5004 DATAAZZURRO,CYN,48,11,159
5005 DATAVIOLA,PUR,64,12,156
5006 DATAVERDE,GRN,80,13,30
5007 DATABLU,BLU,96,14,31
```

```
5008 DATAGIALLO,YEL,112,15,158
5009 DATARANCIO,ORN,128,-1,-1
5010 DATARANCIO CHIARO,LOR,144,-1,-1
5011 DATAROSA,PNK,160,-1,-1
5012 DATACELESTE,LCY,176,-1,-1
5013 DATAVIOLA CHIARO,LPU,192,-1,-1
5014 DATAVERDE CHIARO,LGR,208,-1,-1
5015 DATABLU INTERMEDIO,LBL,224,-1,-1
5016 DATAGIALLO CHIARO,LYE,240,-1,-1
5017 DATAILLECITO,ILL,-1,-1,-1
```

LISTATO COMPLETO DEL PROGRAMMA DIMOSTRATIVO DELLE CAPACITÀ SONORE DEL VIC

```
100 MS=40: NN=12: NO=3: DIM SO%(MS,4),OC$(NO),NO$(NN)
102 NL=5: DIM NL$(NL),NL(NL),SC%(NN,NO)
105 NA=4: FOR I=1 TO 5: SI=36873+I: NEXT I: VO=12
110 FOR I=0 TO NN: READ NO$(I)
120 FOR J=1 TO NO: READ SC%(I,J)
130 NEXT J
150 NEXT I
161 CC$(1)="OTTAVA"
162 CC$(2)="NOTA"
163 CC$(3)="DURATA"
171 AC$(1)="LEGGERE"
172 AC$(2)="SCRIVERE"
173 AC$(3)="COMPORRE"
174 AC$(4)="SUONARE"
181 OC$(1)="B"
182 OC$(2)="M"
183 OC$(3)="A"
190 NL$(0)="": NL(0)=(1/30)*60
191 NL$(1)="1": NL(1)=(1/16)*120
192 NL$(2)="2": NL(2)=(1/8)*120
193 NL$(3)="4": NL(3)=(1/4)*120
194 NL$(4)="8": NL(4)=(1/2)*120
195 NL$(5)="16": NL(5)=1*120
200 PRINT"♥"
300 PRINT"COSA VUOI FARE"
310 INPUT A$
320 FOR I=1 TO NA
330 IF AC$(I)=A$ THEN 400
340 NEXT I
350 PRINT"PUOI SCEGLIERE FRA:"
360 FOR I=1 TO NA: PRINT AC$(I): NEXT I
```

```
370 GOTO 300
400 ON I GOSUB 1100,1200,1300,1400
410 GOTO 200
1100 INPUT"NOME DELLA CANZONE";A$
1110 OPEN 1,1,0,A$
1120 IP=0
1130 FOR I=1 TO 4: INPUT# 1,X
1140 IF X<0 THEN 1180
1150 SO%(IP+1,I)=X: NEXT I
1160 IP=IP+1: IF IP<MS THEN 1130
1170 PRINT"CANZONE TROPPO LUNGA"
1180 CLOSE 1: RETURN
1200 INPUT"NOME DELLA CANZONE";A$
1210 OPEN 1,1,1,A$
1215 IF IP=0 THEN 1230
1220 FOR I=1 TO IP: FOR J=1 TO 4: PRINT# 1,SO%(I,J): NEXT J,I
1230 PRINT# 1,-1: CLOSE 1
1240 RETURN
1300 IP=0
1350 GOSUB 3000
1360 IF OK=1 THEN RETURN
1370 IF IP<MS THEN 1350
1380 PRINT"FATTO. NON C'E' PIU' SPAZIO": RETURN
1400 IF IP=0 THEN RETURN
1410 FOR I=1 TO IP: FOR J=1 TO 3: POKE S(J),SO%(I,J)
1420 NEXT J
1425 POKE S(5),VO
1430 X=TI+SO%(I,4)
1440 IF TI<X THEN 1440
1450 POKE S(5),0
1460 X=TI+NL(0)
1470 IF TI<X THEN 1470
1480 NEXT I
1490 RETURN
3000 CC=1: OK=0: PRINT"] NOTA";IP+1
3010 GOSUB 4000: IF A$="FINE" THEN OK=1: RETURN
3020 IP=IP+1
3040 FOR CN=1 TO NO
3050 IF A$=OC$(CN) THEN 3100
3060 NEXT CN
3070 PRINT"ERRORE. LA ";CC$(CC): PRINT"PUO' ESSERE:"
3080 FOR I=1 TO NO: PRINT OC$(I): NEXT I
3090 IP=IP-1: GOTO 3010
3100 CC=2: GOSUB 4000
3110 FOR I=0 TO NN: IF A$=NO$(I) THEN 3200
3120 NEXT I
```

```
3130 PRINT"NON ESISTE. ESISTONO:";
3140 FOR I=1 TO NN: PRINT NO$(I);" ";: NEXT I: PRINT
3150 GOTO 3100
3200 SO%(IP,2)=SC%(I,CN)
3210 CC=3: GOSUB 4000
3220 FOR I=1 TO NL: IF A$=NL$(I) THEN 3300
3230 NEXT I
3240 PRINT"NON ESISTE. ESISTONO:";
3250 FOR I=1 TO NL: PRINT NL$(I);" ";: NEXT I: PRINT
3260 GOTO 3210
3300 SO%(IP,4)=NL(I)
3310 SO%(IP,1)=0: SO%(IP,3)=0
3999 RETURN
4000 PRINT"BATTI LA ";CC$(CC)
4010 INPUT A$: RETURN
5000 DATA R,0,0,0
5001 DATA DO,135,195,225
5002 DATA DO#,143,199,227
5003 DATA RE,147,201,228
5004 DATA RE#,151,203,229
5005 DATA MI,159,207,231
5006 DATA FA,163,209,232
5007 DATA FA#,167,212,233
5008 DATA SOL,175,215,235
5009 DATA SOL#,179,217,236
5010 DATA LA,183,219,237
5011 DATA LA#,187,221,238
5012 DATA SI,191,223,239
```

Appendice B

TERMINOLOGIA

Se si conoscono certi termini e convenzioni basilari usati nel mondo dei calcolatori numerici, sarà più facile comprendere come questi ultimi funzionano. Parole d'uso comune come "indirizzo" e "locazione", acquistano dei significati particolari se applicate in questo campo. Questa terminologia può essere divisa in due categorie: unità d'informazione e memorizzazione (o ricerca) dell'informazione.

Unità d'informazione Tutte le informazioni elaborate da un computer, siano esse dati o istruzioni, possono essere suddivise in unità d'informazione di varia dimensione. In questa appendice saranno descritte le forme delle unità d'informazione usate dagli elaboratori ed i termini a loro relativi.
Il computer non può lavorare direttamente con i numeri decimali, con le lettere o con i simboli speciali come #, -, +, %, ecc. Come mai, allora, il computer accetta questi caratteri, visto che li adopera sia in ingresso che in uscita?
I componenti elettronici all'interno del computer possono assumere solo due stati. La corrente elettrica può scorrere oppure no; una porzione di nastro magnetico può essere magnetizzata in una direzione oppure in un'altra; un interruttore può essere chiuso o aperto, ecc. È possibile,

quindi, assegnare ad ogni stato un valore. Poiché vi sono solo due stati, comunemente si assegnano ad essi i valori 0 ed 1.
Di conseguenza, tutte le informazioni del computer, siano esse istruzioni, dati, numeri, lettere o simboli speciali, possono essere rappresentati da una combinazione di zero ed uno.
La fig. 7 mostra come una specifica combinazione di zero ed uno possa rappresentare un dato carattere. Il carattere M può essere rappresentato dallo stato di sette circuiti elettronici.
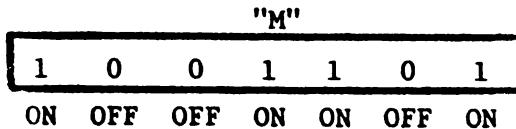
Fig. 7

Ciascun circuito acceso rappresenta un 1, mentre ciascun circuito spento rappresenta uno 0. La combinazione mostrata in figura (1001101) è la rappresentazione della lettera M. Questa rappresentazione è una delle possibili convenzioni. Poiché sono possibili 128 diverse combinazioni con 7 circuiti, è possibile, mediante questi ultimi, rappresentare, 128 caratteri diversi.
Poiché un computer usa solo due cifre (lo zero e l'uno), è d'uso riferirsi ad esse con il nome di cifre binarie o *bit*, che rappresentano l'informazione minima memorizzabile da un computer. Si ricordi, perciò, che il bit può assumere solo il valore 0 o il valore 1. Poiché i computer non possono gestire il singolo bit nelle operazioni di trasferimento dell'informazio da o verso la memoria, si usano raggruppamenti di bit detti *word* (parola) *byte*. In molti microcomputer 8 bit formano un *byte*.
Un byte può rappresentare un numero decimale, una lettera, un simbolo speciale o parte di un indirizzo. La parola può essere composta da uno o più byte, a seconda del tipo di computer. Molti microcomputer adottano una parola di un byte e quest'ultimo è, appunto, l'elemento minimo di informazione che essi possono indirizzare, che, cioè, possono gestire nelle operazioni di trasferimento dell'informazione da o verso la memoria. Il bit, pur essendo la vera unità minima di informazione, non è indirizzabile. Si ricordi che un byte equivale ad 8 bit.
L'esempio di fig. 7 mostra 7 bit. Il codice ASCII standard adotta 7 bit per la rappresentazione dei caratteri; l'ottavo bit è usato per il riconoscimento degli errori. Un carattere, perciò, utilizza un intero byte.

Memorizzazione e ricerca Ciascuna informazione utilizzata dal computer deve essere memorizzata in una zona della memoria dove successivamente possa essere ritrovata. All'occorrenza, l'informazione dovrà essere localizzata nell'area di memoria e recuperata.

Poiché la memoria principale può memorizzare migliaia di bit di informazioni, essa deve essere organizzata in maniera tale che la CPU (*Central Processing Unit*) sappia dove memorizzare l'informazione e dove ritrovarla per l'elaborazione necessaria.
La memoria principale è costituita da migliaia di locazioni o celle di memoria. Nel caso del VIC, le locazioni sono composte da singoli byte. Ciascuna locazione sarà utilizzata dalla CPU durante l'elaborazione o per memorizzarne i risultati. Ovviamente, ogni locazione di memoria può contenere solo un'informazione per volta. Ad ogni locazione viene assegnato uno specifico indirizzo in fase di progettazione del computer. È importante ricordare che l'indirizzo specifica la posizione di un byte nella memoria, ma non il suo contenuto.
Quando un'informazione viene memorizzata in una data locazione, essa sostituisce il contenuto precedente. Tuttavia, leggere il contenuto di una locazione della memoria non altera il contenuto stesso. Una volta che l'informazione è stata memorizzata, è possibile leggerla più volte. Una locazione è simile alla camera di un hotel dove risiede per un certo tempo l'informazione. Il numero della stanza corrisponde all'indirizzo della locazione.
Gli indirizzi delle locazioni di memoria sono numeri interi da zero in poi, tanti quante sono le locazioni. Poiché ciascuna locazione possiede un proprio indirizzo, la CPU può accedere direttamente a qualsiasi locazione della memoria. Sebbene l'indirizzo della locazione sia sempre lo stesso, il suo contenuto può essere modificato in qualsiasi istante.
Prima che un computer possa effettuare una elaborazione, sono necessari due tipi di informazioni: il dato da elaborare e l'istruzione che indica al computer cosa farne. Sia il dato che l'istruzione sono memorizzati nella memoria principale. La sola differenza tra le istruzioni e i dati è il modo in cui il computer cerca di interpretare l'informazione nel byte dato.
Le memorie dei computer sono disponibili in varie dimensioni. Per dimensione di memoria si intende il numero di byte disponibili. La lettera K è l'unità di grandezza per la dimensione della memoria. Un K equivale a 1024 byte. È importante ricordare che ad ogni locazione di memoria (byte) è assegnato un unico indirizzo. Una memoria di 32 K equivale perciò a 32768 byte ovvero dispone di 32768 indirizzi diversi di memoria.

Simboli

I simboli trasmettono informazioni; pur non essendo l'informazione stessa, essi la rappresentano. I caratteri stampati su questa pagina sono simboli che, una volta compresi, trasmettono al lettore il pensiero dello scrittore.

Il significato dei simboli è definito solo per convenzione o per definizione. Un simbolo può avere un certo significato per certe persone, uno differente per altre e nessuno per altre ancora. Per poter comunicare con un computer, si devono usare i simboli che il computer può riconoscere ed interpretare. La scelta di tali simboli ed il loro significato fa parte di una convenzione adottata dai progettisti di computer. Molti dei simboli usati con il Basic fanno parte di simboli standard usati ogni giorno. Alcuni simboli sono invece stati modificati per potersi adattare ai vari tipi di terminali e per evitare confusioni con altri simboli.

Tabella 2. Simboli aritmetici.

Simbolo	*Significato*	*Esempio*
+	somma	Y + 7
−	sottrae	Y − 7
*	moltiplica	Y * 7
/	divide	Y / 7
↑	eleva alla potenza di	Y ↑ 7

1. *Simboli aritmetici* La tabella 2 definisce i più comuni simboli aritmetici usati. I simboli per l'addizione, la sottrazione e la divisione sono gli stessi usati in aritmetica.
Si noti che, invece, il simbolo della moltiplicazione è un asterisco al posto del solito ×.
Si noti, inoltre, che il simbolo ↑ indica l'elevazione a potenza. Come indicato in tabella, 2↑4 è uguale a 16. Si parla in questo caso anche di esponenziazione poiché è simile a piazzare un esponente dopo il numero. Per esempio, l'espressione 2↑4 può essere letta sia come "due elevato alla quarta", sia come "due con esponente quattro". In entrambi i casi la funzione della potenza (o esponente) è la stessa. Essa indica quante volte la base deve essere moltiplicata per se stessa. Cioè, 2↑4 indica che il numero due deve essere moltiplicato per se stesso per quattro volte.

2. *Simboli di relazione* I computer sono spesso usati per confrontare due quantità tra loro per determinare se sono o meno uguali, oppure se l'una è maggiore o minore dell'altra. I simboli usati per esprimere queste condizioni sono elencati in tabella 3.
In Basic il segno = ha due interpretazioni. Il tabella 3 il segno = ha significato condizionale: A = B. Questo significa controllare una condizione di uguaglianza, cioè, controllare se il valore di A è esattamente uguale a quello di B.

Tabella 3. Simboli di relazione.

Simbolo	*Significato*	*Esempio*	*Commento*
=	uguale a	A = B	A è uguale a
< >	non uguale a	A < > B	A non è uguale a B
>	maggiore di	X > 7	X è maggiore di 7
<	minore di	X < 7	X è minore di 7
< =	minore o	X < = 7	X è minore o uguale a 7
= <	uguale a		
> =	maggiore o	X > = 7	X è maggiore o
= >	uguale a		uguale a 7

Un secondo caso in cui viene adoperato il simbolo = è nella scrittura delle equazioni per programmi. In tale contesto, il significato di = deve essere considerato non come uguaglianza, ma come assegnazione: A = B va letto come "il valore di B si sostituisce al vecchio valore di A".
Si supponga che nella locazione di A sia memorizzato il valore 3 e che in quella di B vi sia il valore 10. L'equazione A = B modifica solo il valore di A. Sia la locazione di A che quella di B avranno lo stesso contenuto (10).
Si osservi che il programmatore ha usato i simboli A e B per indicare delle locazioni, lasciando all'interprete Basic il compito di assegnare i rispettivi indirizzi. Questa operazione è nota con il nome di *indirizzamento simbolico*.

SISTEMI NUMERICI

Quando l'uomo esegue manualmente un'elaborazione delle informazioni, è in grado di maneggiare altrettanto bene numeri o lettere. Un computer invece, essendo una macchina binaria, converte le informazioni in zero ed uno. Perciò un computer deve effettuare tutte le operazioni matematiche usando il sistema numerico binario e convertire tutte le informazioni alfanumeriche in codice binario.
Per fornire al computer le informazioni opportunamente predisposte per l'elaborazione, si deve comprendere come il computer gestirà tali informazioni. Questo paragrafo si occuperà dei numeri binari. Inoltre saranno presentati i sistemi numerici ottale ed esadecimale, che facilitano il trattamento di informazioni binarie.
I quattro sistemi numerici di cui si tratterà sono: il sistema decimale, usato quotidianamente; il sistema binario, che il computer adotta a cau-

sa della semplice implementazione elettronica; il sistema ottale e quello esadecimale, che vengono usati dai programmatori per rappresentare i numeri binari, essendo più facili da usare. I numeri esadecimali (od esa, per brevità) e quelli ottali, tuttavia, devono essere convertiti in binari prima di essere elaborati dal computer.
Si noti come in tabella 4 ciascun sistema numerico sia caratterizzato da un'unica base o radice. Questa base corrisponde al numero di cifre o simboli che vengono usati nel particolare sistema. Per esempio, il sistema decimale adotta 10 cifre (dallo 0 al 9) e, quindi, la sua base è 10.

Tabella 4. Sistemi numerici.

Sistema	*Base*	*Cifre*
Decimale	10	da 0 a 9
Esadecimale	16	da 0 a 9 e da A a F
Ottale	8	da 0 a 7
Binario	2	da 0 a 1

1. *Decimale, binario, ottale, esadecimale* Nel sistema decimale, sono necessarie due o più cifre per rappresentare un numero maggiore di 9. Quando un numero contiene due o più cifre, a ciascuna cifra viene assegnato uno specifico valore detto *posizionale*. Questo valore posizionale è uguale alla base del sistema numerico elevata a certe potenze. Nel sistema decimale, i valori posizionali sono le potenze di 10 (cioè unità, decine, centinaia, migliaia, ecc.).

CONVERSIONE DECIMALE-BINARIA

Applicare il seguente algoritmo:

1. Dividere il numero decimale per 2 e salvare il resto.
2. Dividere il quoziente della precedente divisione per 2 e salvare il resto.
3. Continuare con il passo 2 finché il quoziente non sia zero.
4. I resti delle divisioni rappresentano le cifre del numero binario. Il primo resto è la cifra più a destra o meno significativa (LSD, ovvero *Least Significant Digit*), mentre l'ultimo resto è la cifra più a sinistra o più significativa (MSD, ovvero *Most Significant Digit*) del numero binario.

L'esempio sotto mostra come convertire il numero decimale 38 in binario.

Numero/base	*Quoziente*	*Resto*
38/2	19	0 (LSD)
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1 (MSD)

Quindi: 38 = 100110

Tabella 6.

Decimale Base 10	*Esadecimale Base 16*	*Ottale Base 8*	*Binario Base 2*
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

Nel sistema binario, o sistema in base 2, si hanno solo due simboli: 0 ed 1. I valori posizionali in questo sistema sono le potenze di due; cioè, 1, 2, 4, 8, 16, 32, ecc.
Per determinare il valore di un numero, si deve, innanzitutto, sapere in quale base è rappresentato. Quindi, mediante i numeri nel riquadro precedente, si determina il valore posizionale con quella base. Successivamente, si moltiplica ciascuna cifra del numero per il suo corrispondente valore posizionale. Infine si sommano tutti questi numeri tra loro. Provate a determinare il valore del numero binario 1011.

Moltiplicate ciascuna cifra di 1011 per il suo valore posizionale:

$1 \times 1 = 1$
$1 \times 2 = 2$
$0 \times 4 = 0$
$1 \times 8 = 8$

Ora sommate: $8+0+2+1 = 11$

Il valore del numero binario 1011 è uguale al decimale 11. Il metodo indicato funziona con qualsiasi sistema numerico, usando l'appropriata base anziché 2.
Il sistema esadecimale (per brevità esa) ha base sedici. In base sedici sono necessari 16 simboli diversi. I simboli comunemente usati sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
La tabella 6 indica le cifre per le quattro differenti basi. Si noti che occorrono più cifre per rappresentare un numero, man mano che la base diventa più piccola. Per esempio, il numero 15 richiede 4 cifre binarie, 2 ottali, 2 decimali e solo una cifra esadecimale.
Provate a convertire il numero esa 8AF in decimale. Ricordate che qualsiasi base elevata a zero dà sempre 1, mentre elevata ad 1 dà sempre se stessa.

16 ↑ 0 = 1
16 ↑ 1 = 16
16 ↑ 2 = 256

Questi sono i valori delle posizioni

F×1 = 15×1 = 15
A×16 = 10×16 = 160
8×256 = 2048

F esadecimale è uguale a 15 decimale

Ora sommate i risultati:

$15+160+2048 = 2223$

Il numero esadecimale 8AF = 2223 decimale

2. *Conversioni* È spesso necessario convertire un numero da una base in un'altra:

(a) da binario in ottale;
(b) da binario in esadecimale;
(c) da binario in decimale;
(d) da decimale in binario.

I seguenti paragrafi spiegheranno le tecniche di conversione nel caso binario-ottale e nel caso binario-esadecimale.

CONVERSIONE BINARIO-OTTALE

La conversione dei numeri da binario a ottale è relativamente semplice. 8 decimale = 10 ottale = 2↑3. Questo significa che ogni cifra ottale equivale a tre cifre binarie e viceversa (vedi tabella 11).
La procedura di conversione è la seguente:

1 Raggruppare le cifre binarie in insiemi di tre, partendo da destra.
2 Sostituire, per ciascun gruppo, la cifra ottale equivalente (tabella 11).

Convertite in ottale il numero binario 110101111001.

110	101	111	001	Separate in gruppi di tre
6	5	7	1	Sostituite le cifre ottali

$110101111001_2 = 6571_8$

Tabella 11. Equivalenti ottali-binari

Ottale	*Binario*
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

CONVERSIONE BINARIO-ESADECIMALE

Anche la conversione binario-esadecimale è abbastanza semplice poiché 16 è una potenza di 2 (16 = 2 4). Questo significa che è possibile usare la stessa procedura usata precedentemente, ad eccezione del fatto che gli insiemi devono essere composti da 4 cifre binarie. Usando la tabella 13, convertite in esadecimale lo stesso numero binario dell'esempio precedente.

1101 0111 1001 D 7 9	Separate in gruppi di quattro
$110101111001_2 = D79_{16}$	Sostituite le cifre esa

Tabella 13. Equivalenti esadecimali-binari

Esadecimale	*Binario*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

CODICI DEI CARATTERI

Il metodo o sistema per rappresentare i dati è conosciuto con il nome di *codice*. Nel computer, il codice correla i dati ad un numero fisso di bit. Se si usasse solo un bit per rappresentare ciascun simbolo, le scelte possibili sarebbero limitate a due: un simbolo per lo stato "acceso" ed un simbolo per lo stato "spento". Questo limiterebbe seriamente anche la capacità di comunicazione. Se si aggiungesse un altro bit, si raddoppierebbero il numero di caratteri rappresentabili. Avendosi due scelte (0 ed 1) nel caso di un bit, le scelte salirebbero a 4 nel caso di due bit: 00, 01, 10 ed 11. Se si aggiungesse un altro bit ancora, le possibilità raddoppierebbero nuovamente: 000, 001, 010, 011, 100, 101, 110 ed 111. Si noti

che, per *N* bit, il numero delle possibilità è 2↑*N*.
Il codice ASCII è a 7 bit, come stabilito dai costruttori di computer. Il VIC usa una versione modificata del codice ASCII.

OPERAZIONI LOGICHE

Le operazioni logiche vengono usate per operare delle scelte, controllare la grandezza di parametri e controllare il flusso d'esecuzione di un programma.

ESERCIZIO Determinate quali valori numerici vengono usati nel VIC per rappresentare vero e falso.

```
NEW
100 PRINT"BATTI IL NUMERO";
110 INPUT A
200 IF A THEN PRINT "A E' VERO QUANDO A=";A:GOTO 100
210 PRINT "A E' FALSO QUANDO A=";A: GOTO 100
RUN
BATTI IL NUMERO?
1
A E' VERO QUANDO A=1
BATTI IL NUMERO?
0
A E' FALSO QUANDO A=0
BATTI IL NUMERO?
.001
A E' VERO QUANDO A=1E-03
BATTI IL NUMERO?
-1
A E' VERO QUANDO A=-1
BATTI IL NUMERO?
```

Questo risultato mostra come il VIC interpreti 0 come falso ed un numero diverso da zero come vero. Tuttavia, non tutti i numeri diversi da zero funzionano correttamente come vero. Usando l'operatore NOT, occorre usare il valore -1 come vero. Battete 1E40 per terminare il programma.

ESERCIZIO Determinate quale valore di stringa il VIC usa per rappresentare vero e falso.

```
NEW
100 PRINT"BATTI LA STRINGA";
110 INPUT A$
200 IF A$ THEN PRINT"A$ E' VERA QUANDO A$=";A$:GOTO 100
210 PRINT"A$ E' FALSA QUANDO A$=";A$: GOTO 100
RUN
BATTI LA STRINGA?
ABC
A$ E' VERA QUANDO A$=ABC
BATTI LA STRINGA?
""
A$ E' FALSA QUANDO A$=
BATTI LA STRINGA?
" "
A$ E' VERA QUANDO A$=
BATTI LA STRINGA?
```

Questo mostra che la stringa nulla "" viene interpretata come falso. Qualsiasi stringa con almeno un carattere viene invece considerata vero. Battete RETURN e STOP simultaneamente per terminare questo programma.

1. *Funzione* AND La funzione AND rende in uscita vero se tutti gli ingressi sono veri.
La dichiarazione "Mario *e* Giovanni possiedono un computer" è vera solo se entrambi sono possessori di computer.
È possibile costruire una tabella per descrivere il funzionamento di ogni funzione logica. Questa tabella elenca ogni possibile combinazione di valori in ingresso (binari) e il relativo valore in uscita. Ai valori binari 0 ed 1 è possibile sostituire le parole falso e vero. Ecco perché questa tabella è nota con il nome di "tavola della verità".

Tabella 14. Tavola di verità della funzione AND e dell'ingresso

Input		*Output*
A	*B*	*C*
0	0	0
0	1	0
1	0	0
1	1	1

La tabella 14 è la tavola della verità nel caso di una funzione AND a due ingressi. Sono elencate tutte le possibili combinazioni di ingressi con le relative uscite. Si noti che l'uscita si porta ad 1 solo se entrambi gli ingressi sono veri (o 1). Perciò si può affermare che "l'uscita C è vera quando A *e* B sono veri".
Un altro metodo per descrivere la funzione AND è quello di usare una formula matematica. L'uso di una formula matematica porta ad una compatta e precisa notazione per una più semplice rappresentazione della logica digitale. Usando perciò questa tecnica, la relazione fra ingressi ed uscita del caso precedente diventa:

C = A AND B

ESERCIZIO Visualizzate la tavola della verità delle funzioni AND.

```
NEW
90 PRINT"INPUT OUTPUT"
95 PRINT" A  B  C"
100 FOR I=0 TO 1
110 FOR J=0 TO 1
120 PRINT I;J;I AND J
130 NEXT J
140 NEXT I
RUN
```

2. *Funzione* OR *inclusivo* La funzione OR inclusivo fornisce in uscita vero se uno o più ingressi sono veri. Questa funzione restituisce un vero (o 1) all'uscita quando la condizione logica vero è soddisfatta almeno per un ingresso.
La dichiarazione "Marco *o* Paolo possono programmare" è vera se uno od entrambi sanno programmare.
Questa funzione è chiamata OR inclusivo perché in uscita restituisce vero anche quando tutti gli ingressi sono veri (a differenza dell'OR esclusivo, per cui tale condizione rende in uscita 0 o falso). Allorquando si incontra la funzione OR, ci si riferisce alla funzione OR inclusivo.
La tabella 15 è la tavola della verità nel caso di una funzione OR a due ingressi. Sono elencate tutte le possibili combinazioni di ingressi con le relative uscite.
È possibile usare una formula matematica per indicare tale funzione logica. Ad esempio, una funzione logica OR a due ingressi può essere espressa dalla formula:

C = A OR B

ESERCIZIO Visualizzate la tavola della verità della funzione OR. Modificate il programma precedente.

```
120 PRINT I;J;I OR J
RUN
```

Tabella 15. Tavola di verità della funzione OR a due ingressi

Input		*Output*
A	*B*	*C*
0	0	0
0	1	1
1	0	1
1	1	1

3. *Funzione* NOT (*inverter*) Lo scopo di questa funzione è quello di convertire il valore di ingresso nel suo opposto (complementare). La funzione NOT è spesso chiamata "*inverter*" proprio per questa sua peculiarità. Per esempio, se in ingresso vi è 1, all'uscita si ritroverà 0 e viceversa.
La dichiarazione: "Franco *non* è un ingegnere" è falsa se Franco è un ingegnere ed è vera nel caso opposto.
La tabella 16 è la tavola della verità nel caso di una funzione NOT. Si noti che esiste solo un ingresso con due possibili stati.
Anche in questo caso possiamo usare una formula matematica per rappresentare la funzione NOT:

$$C = \text{NOT } A$$

ESERCIZIO Visualizzate la tavola della verità della funzione NOT. Si potrebbe essere tentati di modificare semplicemente il programma precedente, ma si andrebbe incontro ad una delusione. Infatti, dalla tavola della verità del NOT discende che il vero negato (NOT vero) deve essere falso, mentre il falso negato deve essere vero. È stato usato l'1 per il vero e lo 0 per il falso. Provare con:

```
PRINT NOT 0
-1
```

NOT 0 equivale a -1. Perciò, non è possibile utilizzare l'1 come valore per il vero. Come controprova, il NOT vero deve essere falso.

```
PRINT NOT -1
0
```

Il VIC utilizza lo 0 per il falso ed il -1 per il vero.

Tabella 16. Tavola di verità della funzione NOT

Input	*Output*
A	*C*
0	1
1	0

```
NEW
90 PRINT"INPUT   OUTPUT"
100 FOR I=0 TO -1 STEP -1
110 PRINT ABS(I), ABS(NOT I)
120 NEXT I
RUN
```

OPERAZIONI RELAZIONALI

Le operazioni relazionali determinano la validità di una data relazione, assegnando il valore vero (-1) o falso (0).

ESERCIZIO Determinate i risultati che si ottengono dalle seguenti operazioni relazionali.

```
A=1:  B=2:  X=A<B: PRINT X
-1
PRINT A>B
0
PRINT A<B
-1
PRINT A=B
0
```

Usando il -1 per vero e lo 0 per falso, si nota che il risultato dell'operazione relazionale è il valore logico della relazione.
Nel paragrafo sull'organizzazione della memoria si è imparato che il

computer non può gestire il singolo bit. La più piccola unità indirizzabile è il byte. Le operazioni logiche vengono effettuate contemporaneamente su tutti gli 8 bit. Il VIC gestisce due byte (2×8 bit) come unità per operazione logica.

ESERCIZIO Determinate la grandezza massima delle quantità usate nelle operazioni logiche.

```
NEW
A=1:B=2
X=32767:PRINT(A<B) AND X
32767
X=32768:PRINT(A<B) AND X
?ILLEGAL QUANTITY
 ERROR
```

Le operazioni logiche sono lecite solo per quantità di 16 bit o meno.

Siete interessati ai personal computer?

Su questo argomento, nella collana "il piacere del computer" sono stati pubblicati tra gli altri i seguenti titoli.

32 programmi con il PET
Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. Tutti i programmi sono stati verificati e possono essere eseguiti su ogni tipo di PET.
240 pagine, 12.000 lire, sigla PDC 1

Intervista sul personal computer, hardware
Seicento domande e risposte sul mondo dei personal computer. Questo primo volume, dedicato all'hardware, contiene una introduzione, sotto forma di intervista, ai computer in generale e ai microprocessori in particolare.
240 pagine, 12.000 lire, sigla PDC 2

32 programmi con l'Apple
Il volume contiene 32 programmi in Basic, completamente documentati con listati, esecuzioni di prova, istruzioni per far girare il programma, suggerimenti per variazioni, ecc. Tutti i programmi sono stati verificati e possono essere eseguiti su ogni tipo di Apple II.
248 pagine, 12.000 lire, sigla PDC 3

Microsoft Basic
Un breve manuale di introduzione al Microsoft Basic, il Basic dei personal computer. Oggi il Microsoft Basic, una evoluzione e specializzazione del Basic originale, è di fatto lo standard del Basic per microcomputer.
150 pagine,12.000 lire, sigla PDC 4

Pascal
Questo testo è scritto per coloro che non hanno esperienza di calcolatori o programmazione. Gli argomenti sono organizzati in modo che il lettore possa iniziare a programmare fin dall'inizio.
200 pagine, 12.000 lire, sigla PDC 5

32 programmi con il TRS-80
Trentadue programmi, completamente documentati, pronti per essere eseguiti su un TRS-80 modello I. Il volume comprende i listati, le spiegazioni e i consigli per ulteriori progetti.
240 pagine, 12.000 lire, sigla PDC 6

Intervista sul personal computer, software
In questo secondo volume, dedicato al software, altre centinaia di domande e risposte sul mondo dei personal computer. Contiene una introduzione alla programmazione, ai linguaggi assembler e a quelli evoluti.
200 pagine, 12.000 lire, sigla PDC 7

Imparate il Basic con il PET/CBM
Questo libro è stato progettato per essere utile a chiunque desideri imparare a programmare in Basic avendo a disposizione un PET. Studenti di ogni livello, professionisti e dilettanti possono trarre profitto da questo libro e prendere contatto immediato con il mondo dei personal computer.
250 pagine, 12.000 lire, sigla PDC 8

Troverete questi libri nelle principali librerie, oppure potete ordinarli direttamente alla casa editrice compilando la cartolina qui allegata

SUL RETRO ALTRE INFORMAZIONI

Ho trovato questa cartolina nel libro ....

acquistato in ....

In questo spazio potete scrivere quello che pensate di questo libro.

☐ Desidero ricevere il vostro più recente catalogo

☐ Desidero ricevere i volumi qui indicati:

sigla	titolo	prezzo
	totale	

☐ Pagherò al postino l'importo totale indicato + L. 1.000 quale contributo alle spese di spedizione

☐ Allego assegno o vaglia n. .... per l'importo totale indicato

Il personal computer come professione
Il personal computer vi offre mille opportunità di lavoro: potete scrivere articoli o libri su computer; intraprendere un'attività di consulenza o organizzare un'esposizione locale. Per ognuna di queste attività e per molte altre il libro contiene numerosi consigli per la migliore riuscita.
112 pagine, 9.000 lire, sigla PDC 9

Te ne intendi di computer?
Lo scopo di questo libro è di aumentare il livello della vostra comprensione dei computer. Sapere cosa possono e cosa non possono fare. Sapere qual è il loro ruolo nella società, sapere quali problemi creano. Dopo questa lettura, ve ne intenderete di computer.
144 pagine, 8.500 lire, sigla PDC 10

Il Basic e il personal computer, uno: introduzione
Il libro, scritto in tono amichevole e informale, non richiede precedenti esperienze con i computer. Vi è compresa una presentazione del Basic con decine di esempi dettagliati, che fanno diventare realtà le vostre idee.
190 pagine,14.000 lire, sigla PDC 11

Imparate il linguaggio dell'Apple
Se conoscete già il Basic e possedete un computer Apple, siete sulla strada giusta per diventare un esperto in materia. Questo libro vi mette in grado di comprendere il linguaggio macchina dell'Apple partendo dagli esempi più semplici ed arrivando all'uso del miniassembler.
340 pagine, 15.000 lire, sigla PDC 12

Il Basic e il personal computer, due: applicazioni
Questo volume contiene numerosi programmi in Basic adatti ad ogni personal computer. L'uso delle variabili alfanumeriche, gli algoritmi di sort, i giochi, l'arte con il computer sono esempi di programmi illustrati nel volume.
200 pagine, 14.000 lire, sigla PDC 13

Il manuale del CP/M
Il CP/M è il sistema operativo maggiormente usato per i personal computer. Questo volume è un'introduzione breve ma efficace alla tecnica e alla filosofia del CP/M.
120 pagine, 9.500 lire, sigla PDC 14

A scuola con il PET/CBM
Trenta programmi didattici per ogni tipo di PET/CBM matematica, fisica, statistica, ecc.
160 pagine, 13.000 lire, sigla PDC 15

Il manuale dell'Atom
L'edizione italiana del manuale originale di questo personal computer, ora importato in Italia.
300 pagine, 18.500 lire, sigla PDC 16

Il libro del Commodore VIC 20
Questo libro è per i possessori, i futuri possessori e gli utenti di un personal computer VIC Commodore. È inteso come supplemento al manuale della macchina, e presenta numerose caratteristiche del VIC.
156 pagine,12.000 lire, sigla PDC 17

Il debug nei personal computer
Il "debug", in informatica, è la messa a punto dei programmi, la ricerca e la correzione finale degli errori. Si tratta di una operazione di grande importanza, che si può effettuare in vari modi, tutti trattati in questo libro.
144 pagine, 15.000 lire, sigla PDC 18

Programmazione in Basic per l'uomo d'affari
Questo libro è per gli uomini d'affari che vogliono imparare ad usare il calcolatore. Il libro comincia con la convinzione che sia più facile per un uomo d'affari imparare a programmare, che per un programmatore imparare l'organizzazione e la gestione di una azienda.
232 pagine, 19.000 lire, sigla PDC 19

Il Sinclair ZX-81
È il microcomputer più venduto nel mondo. Ma come ogni computer, necessita di software e documentazione: questo libro contiene 37 programmi che illustrano tutte le caratteristiche e le capacità della macchina.
160 pagine, 13.000 lire, sigla PDC 20

Dal Basic al Pascal
Questa eccezionale guida rende facile per chiunque passare dalla propria conoscenza del Basic ad una perfetta padronanza del Pascal, un linguaggio a 2 livelli che consente di accelerare il tempo reale di operazione e di controllo della programmazione e che richiede uno spazio inferiore di memoria nel vostro computer.
320 pagine, 19.000 lire, sigla PDC 21

cognome e nome

indirizzo

cap. località

(partita iva o codice fiscale)

firma

il formato della cartolina è conforme alle vigenti norme postali

cedola di commissione libraria

franco muzzio editore

via bonporti, 36

35141 padova

*Questo libro è per i possessori, i futuri possessori e gli utenti di un personal computer VIC Commodore.
È inteso come supplemento al manuale della macchina, e presenta numerose caratteristiche del VIC.*

*Gli esercizi passo-passo del libro vi porteranno velocemente al miglior utilizzo del VIC.
Il libro è pratico, e non richiede alcuna conoscenza di matematica e aritmetica; se solo sapete contare, avete la preparazione sufficiente per léggerlo.*

ISBN 88-7021-221-1

franco muzzio & c. editore

L. 12.000 (11.776)

17

David Schultz

Il libro del Commodore VIC 20